# Table Of Contents

## 1 Working with the SDK Library ................................................................. 8

## 2 What's New ................................................................................................ 14

2.1 What's New in the SAP Business ByDesign Studio .................................. 14
2.2 What's New in the SAP Business ByDesign Studio Library ...................... 15
2.3 What's New in the Integrated Development Environment ....................... 16
2.4 What's New in Lifecycle Management ...................................................... 18
2.5 What's New in Administration ................................................................. 18
2.6 What's New in the Developer Desktop .................................................... 19
2.7 What's New in Business Objects and Business Object Extensions ............ 21
2.8 What's New in Business Configuration .................................................... 22
2.9 What's New in Analytics ............................................................................ 23
2.10 What's New in Extensibility ..................................................................... 24
2.11 What's New in Mashups .......................................................................... 25
2.12 What's New in Web Services .................................................................... 26
2.13 What's New in the Syntax for Business Objects and Business Object Extensions .................................................. 27
2.14 What's New in the Syntax for Implementation of Business Logic ............ 28
2.15 What's New in Reuse Libraries and Built-In Functions ............................ 31
2.16 What's New in the Public Solution Model ............................................. 32

## 3 Getting Started .......................................................................................... 33

3.1 Getting Started with the SDK ................................................................. 33
3.2 About the Software Development Kit (SDK) ......................................... 35
3.3 First Steps .................................................................................................. 37
    Installing the Software Development Kit ................................................. 37
    User Roles .................................................................................................. 38
    User Setup Quick Guide .......................................................................... 39
3.4 Introduction to the Integrated Development Environment ....................... 44
    Working in the Integrated Development Environment ............................ 44
    Help Content Installation ........................................................................ 47
    My Solutions ............................................................................................ 53
    Repository Explorer ................................................................................ 53
    Solution Explorer .................................................................................... 56
    Add New Item Dialog Box ........................................................................ 58
    Properties Window ................................................................................... 63
    Trace Explorer ........................................................................................ 65
    Working with the Wizards ........................................................................ 65
3.5 Public Solution Model (PSM) ................................................................... 66
    About the Public Solution Model (PSM) .................................................. 66
4 Administration and Lifecycle Management .................................................. 127
4.1 Overview of Administration and Lifecycle Management .......................... 127
4.2 Lifecycle Management of Customer-Specific Solutions ............................ 127
4.3 Quality Criteria for Customer-Specific Solutions ..................................... 128
4.4 Developing on a Customer’s Tenant ......................................................... 132
  Lifecycle Management for Customer-Specific Solutions on a Customer’s Preproduction Tenant ............... 132
  Create a Patch on a Customer’s Preproduction Tenant ................................ 134
  Lifecycle Management for Customer-Specific Solutions on a Customer’s Test Tenant ......................... 135
  Create a Patch on a Customer’s Test Tenant ............................................. 137
4.5 Developing on Your Development Tenant ............................................... 139
  Lifecycle Management of Customer-Specific Solutions on Your Development Tenant ....................... 139
  Create a Patch on Your Development Tenant ............................................ 141
  Switch a Customer Assignment ................................................................ 143
4.6 Administration .......................................................................................... 144
  Administration Quick Guide for Customer-Specific Solutions ..................... 144
  Implementation Manager Quick Guide for Customer-Specific Solutions .......... 146
4.7 Solutions and Solution Templates ............................................................... 149
  Solution Templates Quick Guide for Customer-Specific Solutions ................ 149
  Solution Templates .................................................................................... 150
  Create a Template for Customer-Specific Solutions ................................... 151
  Create a Customer-Specific Solution .......................................................... 152
4.8 Patches for Customer-Specific Solutions ................................................... 153
4.9 Maintenance Mode .................................................................................... 154
4.10 Upgrade Information for Customer-Specific Solutions ............................ 154
5 Scripting Languages ..................................................................................... 156
5.1 Overview of Scripting Languages ............................................................. 156
5.2 Scripting Language for the SDK ............................................................... 156
  Scripting Language Reference .................................................................... 156
  Business Object Definitions ....................................................................... 158
  Business Object Extension Definitions ....................................................... 170
  Business Logic ............................................................................................ 178
  Reuse Libraries .......................................................................................... 202
  Basic Reuse Libraries .................................................................................. 202
6 Developer Desktop ......................................................... 260
   6.1 Overview of the Developer Desktop ................................... 260
   6.2 General Topics .......................................................... 261
       Log On to the Repository ............................................... 261
       Report an Incident ..................................................... 262
       Translation .............................................................. 263
       Business and Technical Background .............................. 265
       Tasks .................................................................... 269
   6.3 Business Configuration ................................................ 272
       Business Configuration Quick Guide ................................ 272
       Business and Technical Background ................................ 277
       Tasks .................................................................... 269
   6.4 Business Objects .......................................................... 297
       Business Objects Quick Guide ....................................... 297
       Business and Technical Background ................................ 301
       Tasks .................................................................... 307
   6.5 Analytics .......................................................... 323
       Analytics Quick Guide .................................................. 323
       Business and Technical Background ................................ 327
       Tasks .................................................................... 336
   6.6 Extensibility .............................................................. 342
       Business Object Extensions .......................................... 342
       Tasks .................................................................... 345
       Process Extension Scenarios ......................................... 355
       Tasks .................................................................... 358
       Enhancement Implementations ..................................... 364
   6.7 Mashups .......................................................... 368
       Mashups Quick Guide ...................................................... 368
       Business and Technical Background ................................ 369
       Tasks .................................................................... 382
   6.8 Print Forms .............................................................. 411
       Print Forms Quick Guide ............................................... 411
       Associating an .xdp File with Adobe LiveCycle Designer ...... 413
       Create a Print Form ..................................................... 413
   6.9 Service Integration ......................................................... 414
Service Integration Quick Guide ............................................................ 414
Condition Evaluation for Internal Communication ............................... 416
Tasks .......................................................................................... 418
6.10 Tracing .................................................................................. 424
Tracing Quick Guide ....................................................................... 424
Business and Technical Background .................................................. 425
Tasks ....................................................................................... 429
6.11 User Interface Custom Panes and Custom Controls ......................... 432
Custom Panes and Custom Controls Quick Guide ................................. 432
Business and Technical Background .................................................. 433
SDK Namespace ........................................................................... 435
Common.Controller Namespace ......................................................... 441
Common.Controller.Data Namespace .................................................. 443
Common.Controller.Event Namespace ................................................ 446
Common.Controller.Navigation Namespace ......................................... 449
Common.Controls Namespace ............................................................ 452
Controller Namespace .................................................................. 455
Controller.Data Namespace ............................................................... 456
Controller.Property Namespace ......................................................... 465
Controls Namespace ................................................................... 470
Model Namespace ....................................................................... 471
Tasks ....................................................................................... 472
Example: Best Employee Graph ......................................................... 476
6.12 Web Services .......................................................................... 480
Web Services Quick Guide ................................................................. 480
Business and Technical Background .................................................. 483
Tasks ....................................................................................... 494
Sample Code: Event BeforeSave Web Service Operation ....................... 506

7 User Interface Designer .................................................................. 508
7.1 Overview of the User Interface Designer ........................................... 508
7.2 Getting Started with the User Interface Designer ............................... 508
Introduction to the User Interface Designer .......................................... 508
Features of the User Interface Designer .............................................. 509
7.3 Modeling Floorplans ................................................................ 512
Floorplan Modeling Quick Guide ......................................................... 512
Tasks ....................................................................................... 515
7.4 Modify Your Floorplans ............................................................... 532
Floorplan Modification Quick Guide .................................................. 532
Business and technical background .................................................... 535
Tasks ....................................................................................... 540
7.5 Modify an SAP Floorplan ............................................................ 562
Change Transactions Quick Guide ..................................................... 562
Business and technical background .................................................... 564
Tasks ....................................................................................... 565
7.6 User Interface Modifications for Mobile Applications ....................... 574
8 Style Guides and Reference Guides .................................................. 584
8.1 Overview of Style Guides and Reference Guides .......................... 584
8.2 Style Guides ............................................................................. 584
   Style Guide for Writing System Message Texts ............................ 584
   Standards for Errors and Failures .............................................. 589
   Standards for Inability to Perform Action, Status and Authorization Problems ........................................ 590
   Standards for Existence and Availability of Objects, Missing Data, and Search Results ...................... 592
   Standards for Invalid Data and Value Problems ........................... 594
   Standards for Comparison Problems .......................................... 596
   Standards for Object Selection ................................................... 597
   Standards for Application Log Messages .................................... 597
   General Standards for Message Long Texts ................................. 602
8.3 Reference Guides ...................................................................... 609
   Reference Guide for the SAP Solution Documentation ................. 609

9 Glossary ....................................................................................... 623

10 SAP - Copyrights and Trademarks ............................................... 656

11 Important Disclaimers on Legal Aspects ..................................... 658
1 Working with the SDK Library

In this library you can view reference documentation about the software development kit (SDK) for SAP Business ByDesign. The library contains reference documentation to help you get started with the SDK, as well as information and templates related to quality assurance, information for SDK administrators, and detailed information on development activities in the SDK. The library also contains reference documentation for the user interface designer (UI designer), and references the general concepts and tools in the SAP solution.

You can access the library as follows:

- In the studio, select Help SAP Business ByDesign Studio Help. The library opens in your Web browser and is displayed using a content player. This option does not offer a function to search the library.
- In the studio, select Help Manage Help Settings. You can install the help content manually and view the content using Microsoft Help Viewer, which includes a search function. For more information, see Help Content Installation [page 47].
- If you install the help content using Microsoft Help Viewer, you can also access context-sensitive help for wizards, dialog boxes, and tool windows by pressing F1 or clicking the Help icons on the UI.

This documentation does not describe the standard features and technologies in Microsoft Visual Studio. For information about Microsoft Visual Studio, see http://www.microsoft.com.

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages caused by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.
Getting Started

<table>
<thead>
<tr>
<th>First Steps</th>
<th>Introducing the SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learn about the following:</td>
<td>Learn about the following:</td>
</tr>
<tr>
<td>- <strong>Installing the Software Development Kit</strong> [page 37] You have to install the latest version of the software development kit (SDK) from the SAP Service Marketplace. This document takes you through the requirements and steps to install the SDK. <strong>IMPORTANT:</strong> Prior to working in the SDK, you must change the initial password provided to you in the SAP solution.</td>
<td>- <strong>Features of the SDK</strong> The SDK enables you to adapt and enhance the solution capabilities of the SAP solution. Using the SDK, you can develop, deploy, and test specific add-on functionality. For an overview of the SDK features, see <strong>About the Software Development Kit (SDK)</strong> [page 35].</td>
</tr>
<tr>
<td>- <strong>User Setup Quick Guide</strong> Before you can begin working in the SDK, the initial user has to create the required users for the SDK and for the SAP solution. This quick guide explains the entire process from creating service agents, requesting users for these service agents, assigning an ID and an initial password to each user, and assigning the required work centers. In addition, it also explains the process to create business users for testing purposes in the SAP solution. For more information, see <strong>User Setup Quick Guide</strong> [page 39].</td>
<td>- <strong>Working in the Integrated Development Environment</strong> [page 44] The development tools provided by the SDK are presented in an integrated development environment (IDE) based on Microsoft Visual Studio. The user interface (UI) of the SDK is referred to as the developer desktop.</td>
</tr>
<tr>
<td>- <strong>User Roles</strong> [page 38] Find out about the different user roles in the SDK: ○ For development purposes: SDK Administrator and Developer ○ For testing purposes: Business User</td>
<td>- <strong>Working with Wizards</strong> [page 65] The wizards guide you through a logical sequence of steps, making it easier and quicker for you to perform activities that are complex, unfamiliar, or infrequently performed. All wizards are easy to access and follow similar user interface patterns.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Get an Introduction to Business Object Modeling and the Public Solution Model</th>
<th>Learn How to Build Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Introduction to Business Object Modeling</strong> [page 70] The SAP solution is built using the concept of business objects to model the business environment.</td>
<td>- <strong>Example: Building a Solution</strong> [page 88] In this example you build a solution for the SAP solution to manage the allocation of car parking spaces to employees.</td>
</tr>
<tr>
<td>- <strong>Public Solution Model</strong> The public solution model contains the released content that can be used in the SDK. The information about the released content types is available in the Business Center at <a href="https://www.sme.sap.com">https://www.sme.sap.com</a> under [SAP Business ByDesign ➤ Community ➤ Wiki Info Exchange ➤ SAP Solutions On Demand Studio ➤ in the section <a href="https://www.sme.sap.com">Partner/Developer Information ➤ Public Solution Model</a> Here you can find information about released business objects, embedded components, object value selectors, UI imports, UI outlets, and data types.</td>
<td>- <strong>Enablement Service</strong> SAP provides an SAP Business ByDesign Studio Experience enablement service. Further details related to curriculum, prerequisites, pricing, delivery format, schedule, and registration can be found in the Business Center at <a href="https://www.sme.sap.com">https://www.sme.sap.com</a> under <a href="https://www.sme.sap.com">SAP Business ByDesign ➤ Learn ➤ Enablement Services ➤ ByD Studio Experience</a></td>
</tr>
<tr>
<td>- <strong>Test Shell Quick Guide</strong> [page 85] Learn how the Test Shell work center view in the SAP solution enables you to browse the SAP business objects and entities that you can use in developing your solution. <strong>NOTE:</strong> The Test Shell functionality is only available to partners in their partner development tenants. The functionality is not available in customer systems or customer-specific development systems.</td>
<td></td>
</tr>
</tbody>
</table>
Check Out the Development and Mobile Development Guidelines

Check out the guidelines in the Business Center at https://www.sme.sap.com under SAP Business ByDesign Community Wiki Info Exchange SAP Solutions OnDemand Studio in the section Partner/Developer Information under Topics in Detail Documentation.

- Development Guideline
  The Development Guideline provides general programming rules and guidance on selected product standards.

- Mobile Development Guideline
  The Mobile Development Guideline provides general UI guidelines related to mobile development.

Find Out How You Can Localize Your Solution

Check out the Localization Toolkit [page 125]. The localization toolkit is a set of development tools and instructions that enable SAP partners to extend and adapt the capabilities of the SAP Business ByDesign solution to provide a locally-compliant solution for customers in countries where a fully-localized SAP country version is not available. You can access the toolkit in the Business Center at https://www.sme.sap.com under SAP Business ByDesign Community Wiki Info Exchange Country Information Customer-Specific Localization Localization Toolkit.

Administration and Lifecycle Management

Get Informed About Administration and Lifecycle Management of Customer-Specific Solutions

- Administration Quick Guide for Customer-Specific Solutions [page 144]
  The Administration toolbar provides access to a number of administrative tasks, such as creating a new solution, deleting a solution, and managing user sessions.

- Implementation Manager Quick Guide for Customer-Specific Solutions [page 146]
  You use the Implementation Manager to manage the lifecycle of customer-specific solutions.

- Lifecycle Management of Customer-Specific Solutions [page 127]
  The SDK supports the full end-to-end lifecycle management of customer-specific solutions. This includes developing, testing, and deploying your solution, as well as creating patches to deliver updates to your customer.

- Create a Customer-Specific Solution [page 152]
  Learn how to create solutions for specific customers.

- Create a Solution Template [page 151]
  Learn about how to create templates for customer-specific solutions.

- Patches for Customer-Specific Solutions [page 153]
  If you need to make changes or corrections to a customer-specific solution after you have assembled it, you can create a patch. Depending on the tenant on which you are working, the process of creating a patch is different.
## Scripting Languages

### Get Informed About Scripting Languages

Check out the reference material for:
- **Scripting Language** [page 156]
  - The scripting language is used to define the following:
    - **Business objects** [page 158]
    - **Business object extensions** [page 170]
  - It is also used to implement the business logic [page 178] for partner solutions. The methods provided for the implementation of the business logic are supplemented by the following built-in functions and reuse libraries:
    - **Built-In Functions** [page 228]
    - **Reuse Libraries** [page 202]
    - **Reuse Libraries for Business Areas** [page 223]
  - The reuse libraries provide basic functions that are used quite often, for example, the retrieval of context data, such as the current date or time of the current identity UUID.

- **Front-End Script for the User Interface Designer** [page 240]
  - The front-end script is the programming language used in the UI designer to define a set of rules and the logic of the behavior of UI controls. For example, you can write a front-end script to enable an edit button only when a field is selected on the user interface.

- **Application Programming Interface for SDK Custom Panes and Custom Controls** [page 433]
  - Learn about the API that enables you to access **Controller.DataContainer** to read the data from the actual running client component and register to events in case of changes. It also enables you to visualize data on your UI in the manner you choose.
# Developer Desktop

## Choose a Help Topic for the Developer Desktop

### General Topics
- **Log On to the Repository** [page 261]
  Find out how to connect to the repository.
- **Report an Incident** [page 262]
  If you encounter a problem during development, you can report an incident in the SAP solution to request help in solving the problem.
- **Translation** [page 263]
  Find out how you can translate your solution content into the languages supported by the SAP solution.

### Business Configuration [page 272]
Learn how to create and change business topics, business options, and business configuration sets for your solution. Also find out how to configure a business configuration view and assign it to the Business Configuration work center, how to check your content in the business adaptation catalog, and how to create, define, and complete implementation project templates. In addition, find information on the tools available in the SAP solution to assist you in creating and editing your business configuration content.

### Business Objects [page 297]
Learn how to create a business object, define its nodes and relevant elements, create script files, activate your business object, and generate the screens for your business object.

### Extensibility
- **Business Object Extensions** [page 342]
  Learn how to create a business object extension, define business logic for a business object extension, and add extension fields to forms, screens, reports, and the Enterprise Search. In addition, find information on the tools available in the SAP solution to assist you with extensibility.

### Analytics [page 323]
Learn how to create and change data sources, and create and manage reports, and get informed about analytical report content, transformation rules, and aggregation. In addition, find information on the tools available in the SAP solution to assist you in defining your reports.

### Mashups [page 368]
Learn about creating mashups using the SDK and the UI designer.

### Print Forms [page 411]
Learn how to create a print form, add a preview button to it, and test the print form. In addition, find information on the tools available in the SAP solution to assist you in creating and editing your print forms.

### Service Integration [page 414]
Learn how to configure internal communication to communicate with business objects in different deployment units, and create XML file input to import external XML data into a business object. In addition, find information on the tool for file input in the SAP solution.

### Tracing [page 424]
Learn how to use tracing to record information about your code as it is being executed. From the Trace Explorer, you can run a local trace, run a trace for a business user in the SAP solution, use incident traces for troubleshooting purposes, and view the results of a recorded trace.

### Custom Panes and Custom Controls [page 432]
Learn how to create custom panes and custom controls, create embedded components for custom panes, and add embedded components to a screen. Check out the example on creating a custom pane and uploading it into an embedded component in the SDK.

### Web Services [page 480]
Learn how to create, modify, authorize and test a Web service, and how to remove authorization to a Web service.

## User Interface Designer

### Get Informed About the UI Designer

© 2012 SAP AG. All rights reserved. • PUBLIC
Introduction to the UI Designer  [page 508]
The user interface designer (UI designer) is integrated into the SDK, and you can use it to create new screens, or modify the screens provided by SAP. The UI designer can also be used to design screens for use on mobile devices.

Features of the UI Designer  [page 509]
Get a quick overview of the common features available to develop user interface.

Application Programming Interface for SDK Custom Panes and Custom Controls  [page 433]
Learn about the API that enables you to access Controller.DataContainer to read the data from the actual running client component and register to events in case of changes, as well as visualize data on your UI in the manner you choose.

Floorplan Modeling  [page 512]
Get an overview of the steps involved in modeling different types of floorplans.

Floorplan Modification  [page 532]
Get informed about the modifications and enhancements you can make to the floorplans you created.

Change Transactions  [page 562]
Get an overview of the changes you can make to SAP floorplans. These changes are termed as change transactions and need anchors to be executed.

Mobile Floorplans  [page 574]
Get an overview about how to adapt floorplans for use on mobile devices and the steps involved in configuring navigation between floorplans.

Style Guides and Reference Guides

<table>
<thead>
<tr>
<th>Style Guides</th>
<th>Reference Guides</th>
</tr>
</thead>
<tbody>
<tr>
<td>[page 584]</td>
<td>[page 609]</td>
</tr>
<tr>
<td>This style guide provides you with the standards and guidelines for writing system messages to be displayed on the user interface (UI) of your solution. It also provides the standards and guidelines for writing system messages that are relevant for SAP Support; these messages are not displayed on the UI. In addition, this style guide provides guidance on writing long-text documents providing additional information to a system message that helps the user to solve very complex error situations. These long-text documents can be incorporated into the Solution Documentation.</td>
<td>This reference guide provides an overview of the main reference documentation types specific to the SAP Business ByDesign solution.</td>
</tr>
</tbody>
</table>

Discussion Forum and Best Practices

<table>
<thead>
<tr>
<th>Get Assistance from Your Peers</th>
<th>Check Out Best Practices for Implementation</th>
</tr>
</thead>
</table>
2  What's New

2.1  What's New in the SAP Business ByDesign Studio

This document provides information about what is new in the SAP Business ByDesign studio Feature Pack 4.0 for customer-specific solutions.

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages caused by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

What Is New and What Has Changed: Topic-Specific

The following topics are new to the entire application:

- **Exceptional Cases for Scalable Solutions**
  The creation of scalable solutions is not supported any longer and only possible in exceptional cases. You should always create customer-specific solutions. However, the creation of scalable solutions is possible in exceptional cases. If you think you require a scalable solution, proceed as follows:
  1. Create an incident and ask SAP Support to put it on component AP-RC-BDS-LM.
  2. Enter the following information:
     - Solution name — use alphanumeric characters and underscores in the name
     - Solution description
     - Solution type (mashup, integrated solution, lightweight solution, add-on, or micro-vertical solution)
     For information about the types available, see https://wiki.sme.sap.com/wiki/x/aIKKEw.
     - Deployment unit of the solution — you can view the available deployment units in the Repository Explorer
     - Detailed explanation as to why you require a scalable solution

  If SAP agrees to your request, you will receive instructions on how to proceed in the studio.

- **User ID for the Software Development Kit (SDK)**
  When setting up the users for the SDK, it is no longer necessary for the user ID to match exactly the technical ID. Instead you can overwrite the user ID proposed by the system, for example, by entering the actual name of the user. For more information, see User Setup Quick Guide [page 39] under Tasks ➔ Assign a User ID and a Password to the Service Agent (for Development Users).

For information about what is new in a specific area in the studio, see:

- What’s New in the SAP Business ByDesign Studio Library [page 15]
- What’s New in the Integrated Development Environment (IDE) [page 16]
- What’s New in the Public Solution Model (PSM) [page 32]
- What’s New in Administration [page 18]
2.2 What's New in the SAP Business ByDesign Studio Library

This document provides information about what is new in the documentation library for the SAP Business ByDesign studio Feature Pack 4.0.

What Is New

The following functions are new.

- **Context-Sensitive Help (F1) for Wizards and Dialog Boxes**
  You can access context-sensitive help for wizards, dialog boxes, and tool windows by pressing `F1` when you open a wizard or dialog box or on selection of a tool window. You can also click the Help icon located in the top right of the user interface element.
  To view context-sensitive help, you must install the Help content manually on your local computer.
  For information about installing and viewing Help content, see [Help Content Installation](#) [page 47].
  You can also access information about installing and viewing Help content in the SAP Business ByDesign studio from the **View** menu by selecting **Help Content Installation**.

- **Context-Sensitive Help (F1) for Keywords (Business Logic)**
  You can access context-sensitive help for keywords when you implement the business logic for your solution in script files. To do this, in the code editor, place your mouse cursor on a keyword and press `F1`.
  Context-sensitive help is available for the following keywords:
  - `as`
  - `break`
  - `case`
  - `collectionof`
  - `continue`
  - `default`
  - `elementsof`
  - `else`
  - `foreach`
  - `if`
  - `import`
  - `in`
  - `raise`
  - `return`
  - `switch`
If documentation is not available for a keyword, when you press F1, an overview document opens (Syntax for Implementation of Business Logic [page 178]).

To view context-sensitive help, you must install the Help content manually on your local computer.

For information about installing and viewing Help content, see Help Content Installation [page 47].

You can also access information about installing and viewing Help content in the SAP Business ByDesign studio from the View menu by selecting Help Content Installation.

- **Context-Sensitive Help (F1) for Keywords (Business Object Definitions)**
  When you define a business object for your solution and press F1 in the code editor, an overview document opens (Syntax for Business Object Definitions [page 158]) that contains a list of functions and keywords with links to further information.

### What Has Changed

The following functions have changed:

- **Documentation on Maintenance of Business Objects and Business Object Extensions**
  There are two new documents detailing the change restrictions for business objects and business object extensions. For more information, see Maintenance of Business Objects [page 305] and Maintenance of Business Object Extensions [page 344].

- **Terminology**
  - SDK Administrator
    The term **SDK administrator** replaces the term **administrator** to differentiate between the administrator user in the SDK and the key user in the SAP solution.

- **Renaming of Business Center Pages**
  The Wiki pages related to SAP Business ByDesign studio have been renamed **SAP Solutions OnDemand Studio**. You can navigate to the pages as follows: [https://www.sme.sap.com](https://www.sme.sap.com) under **SAP Business ByDesign > Community > Wiki Info Exchange > SAP Solutions OnDemand Studio** or by using the tiny link: [https://wiki.sme.sap.com/wiki/x/PIcqCw](https://wiki.sme.sap.com/wiki/x/PIcqCw).

### 2.3 What's New in the Integrated Development Environment

This document provides information about what is new in the integrated development environment (IDE) in the SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

### What Is New

The following functions are new.

- **Name of Repository View Has Changed to My Solutions**
  The name of the **Repository View** tool window has changed to **My Solutions** and the **SAP Content** folder has been removed. You can now view SAP content and the documentation of specific entities that are released with the public solution model (PSM) in the **Repository Explorer**. For more information, see **My Solutions** [page 53] and **Repository Explorer** [page 53].
- **Repository Explorer**
The *Repository Explorer* tool window displays SAP content that is released with the public solution model (PSM) such as business objects, data types, enhancement options, and inbound service interfaces. You can explore the content to gain an understanding of the repository structure and the business context and use of SAP content, which you can reuse and enhance in your solution.

To display SAP content in the *Repository Explorer*, you must log on to the repository. You can open the *Repository Explorer* from the *View* menu by clicking **Other Windows** → *Repository Explorer*.

You can dock the tool window on the edge of the developer desktop or as a tabbed document. For best readability, we recommend that you dock the *Repository Explorer* as a tabbed document. For more information, see *Repository Explorer* [page 53].

- **More Information Available in the Properties Window**
The *Properties* window displays information about the item in your solution that is currently selected in the *Solution Explorer*. When you select the solution, you can view solution properties such as the solution status. When you select an item such as a business object or a UI component, you can view item properties such as the activation status of runtime objects.

You can access the *Properties* window from the *View* menu by clicking *Properties Window* or by pressing **F4**. For more information, see *Properties Window* [page 63].

- **Administration Menu**
You can access existing administration and lifecycle management functions from the *Administration* menu. The *Administration* menu also gives you access to most of the functions provided by the *My Solutions* tool window.

The functions available on the *Administration* menu are:
- **Log On**
- **Log Off**
- **Create Solution**
- **Delete Solution**
- **Session Administrator**
- **Implementation Manager**
- **Admin Log**
- **Switch Customer...**
  This function is new in the SAP Business ByDesign studio Feature Pack 4.0. For more information, see *Switch a Customer Assignment* [page 143].
- **Options and Settings** (also available via the *Tools* menu → *Options*)

For more information about these functions, see *Working in the Integrated Development Environment* [page 44].

- **Add New Item Button**
You can open the *Add New Item* dialog box by clicking the button at the top of the *Solution Explorer* tool window.

Add New Item Button
2.4 What's New in Lifecycle Management

This document provides information about what is new in lifecycle management in the SAP Business ByDesign studio Feature Pack 4.0 for customer-specific solutions.

What Is New

The following functions are new:

- **Maintenance Mode**
  The new maintenance mode has been introduced to customer-specific solutions. A customer-specific solution is in maintenance mode if it has been assembled and downloaded. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer’s production tenant.
  For more information, see Maintenance Mode [page 154].

- **Developing on a development tenant**
  You can now develop your customer-specific solutions on a development tenant in your company or organization, for example, if your customer does not have a test tenant. On the development tenant you can develop solutions for different customers. When you do this, you need to ensure that each solution you create is assigned to the correct customer. For more information, see Switch a Customer Assignment [page 143].
  The finalized solution can be downloaded from the development tenant and can then be uploaded and activated on the customer’s production tenant. For more information, see Lifecycle Management on a Development Tenant [page 139].

2.5 What's New in Administration

This document provides information about what is new for administrators in the SAP Business ByDesign studio Feature Pack 4.0 for customer-specific solutions.

What Is New

The following functions are new:

- **Create Solution Dialog Box**
  The following function is new in the Create Solution dialog box:
    - Create Solution Template
      You can now use the dialog box to create a solution template, which is not assigned to a specific customer. You can use this template as a container for reusable items and import these items into existing customer-specific solutions. This enables you, for example, to rapidly start the development of customer-specific solutions by reusing common development content, for example, for a specific industry.
      The following restrictions and guidelines apply:
• You cannot create a template on a preproduction tenant.

• A template always has the status In Development. You cannot create a patch of a template.

• You are not able to define content using the key user tools, that is, you cannot enable the key user mode for a template.
  For information about the key user mode, see Solution Explorer [page 56] under Solution.

• You cannot create BC sets using SAP BCOs in a template.
  For more information, see Create a Solution Template [page 151].

• Switch Customer
  If you are developing customer-specific solutions on a development tenant, you can create solutions for more than one customer. To do this, you need to ensure that each solution that you create is assigned to the correct customer. For more information, see Switch a Customer Assignment [page 143].

• Implementation Manager
  The following functions are new in the Implementation Manager:
  ○ Copy Solution
    You can use a copy of your solution or template, for example, to continue developing in order to try out different scenarios without disrupting the original. To do this, you can download a copy of an existing solution or template and then upload this copy to the same tenant or a different tenant. For more information, see Implementation Manager Quick Guide for Customer-Specific Solutions [page 146] under Create a Copy of a Solution.
  ○ Import Solution Template
    After you have downloaded a solution template, you can import this template to add its items to any of your customer-specific solutions. For more information, see Solution Templates Quick Guide [page 149] under Import Items of a Solution Template.

What has Changed

The following functions have changed.

• Implementation Manager
  The following functions have changed in the Implementation Manager:
  ○ Assemble and Download Button
    You can now use the Assemble and Download button in the Implementation Manager to download a solution template in order to upload its items to a particular customer-specific solution. For more information, see Solution Templates Quick Guide [page 149] under Download a Solution Template.

2.6 What's New in the Developer Desktop

This document provides information about what is new in the developer desktop in the SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

What Is New

The following functions are new.

• Asynchronous Message-based Communication
The SAP Business ByDesign studio now provides functions that you can use to create asynchronous message-based communication between a business object and another communication partner, such as a different system. For more information, see Service Integration Quick Guide [page 414].

You can use the following functions to support asynchronous message-based communication in your customer-specific solutions:

- Internal Communication
  You can use internal communication to exchange data between business objects that are located in different deployment units (DUs). For more information, see Configure Internal Communication [page 418].

- XML File Input
  In SAP Business ByDesign, key users can use an XML file to import third-party data from other systems into a business object. To enable this functionality, you create an XML schema definition in the SAP Business ByDesign studio. This schema definition allows customers to create an XML file, which can then be used to import the data into the SAP Business ByDesign system. For more information, see Create XML File Input [page 420].

- Error Handling
  You can predefine the business tasks that a key user can monitor in SAP Business ByDesign if errors or conflicts occur during asynchronous communication. To do this, you define these tasks in the studio and assign them to the appropriate messages of the target business object. For more information, see Define Service Integration Error Handling [page 421].

- Condition Evaluation for Internal Communication
  Condition evaluation is used to determine when a business object needs to send a message to the target business object, for example, a message to create or change a business object instance. For each business object identified as relevant for the subsequent service integration process, the system can evaluate a start condition, change condition and a cancel condition. For more information, see Condition Evaluation for Internal Communication [page 416] and Implement the Condition Evaluation Logic for Internal Communication [page 423].

- Translation Support
  The SAP Business ByDesign studio now provides functions that enable the translation of business objects, business configuration content texts, and on-screen texts of other content types into one of the languages supported by SAP Business ByDesign. The translation functions enable you to adapt a customer-specific solution created using the studio for a specific country or region.

  You can access the following functions to support the translation of customer-specific solutions by right-clicking a solution or an item:
  - Export Text for Translation to export on-screen texts and business configuration content in XML Localisation Interchange File Format (XLIFF) for translation into one of the languages supported by SAP Business ByDesign.
  - Import Translation to import a translated XLIFF file.
  - Check Translation Status to monitor the translation status of translatable text strings.

  For more information on translation of texts in the SDK, see Translation Quick Guide [page 263].

- Test Shell
  The Test Shell functionality in the SAP solution is now also available to partners and resellers who are developing customer-specific solutions on a development tenant or a customer’s tenant. The functionality is not available on a customer’s production tenant after go-live. For more information, see Test Shell Quick Guide [page 85].

See Also
What’s New in Business Configuration [page 22]
2.7 What's New in Business Objects and Business Object Extensions

This document provides information about what is new in business objects and business object extensions in the SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

What Is New

The following functions are new:

- **Documentation on Maintenance of Business Objects and Business Object Extensions**
  There are two new documents detailing the change restrictions for business objects and business object extensions. For more information, see [Maintenance of Business Objects](#) and [Maintenance of Business Object Extensions](#).

- **Inclusion of Business Objects in Enterprise Search**
  You can make a custom business object available in search queries, that is, you can include a custom business object in the enterprise search. To do this, in the Solution Explorer, you right-click your custom business object and select Create Query; then in the Query Wizard, under Enterprise Search Details, you define the following fields:
  - **Key Field**
    The field of the business object to be used in search queries to sort or filter search results by ID.
  - **Description Field**
    The field of the business object to be used in search queries to filter search results by description.
  - **Last Changed Field**
    The field of the business object to be used in search queries to sort or filter search results by last change date

  For more information, see [Create a Query](#).

- **AfterLoading Event (only relevant for business objects)**
  The AfterLoading event for filling transient fields in a business object has been introduced. The data in transient fields is not persistent, that is, the data is not saved in the database. For more information, see [Actions, Events and Validations](#).

- **Translation Support for Business Object Labels and Tooltips**
  In the Solution Explorer, when you right-click a business object, a dropdown menu provides the following functions for translation:
  - Export Text for Translation
  - Import Translation
  - Check Translation Status

  This makes it possible for you to translate texts that are used in business objects that you have created. For information on translation of texts in the SDK, see [Translation Quick Guide](#).

- **Search for Business Objects in Add New Item Dialog Box**
You can now search for business objects when you want to create new screens in the Solution Explorer using the Add New Item dialog box. You can search both for custom business objects and for SAP business objects. To do this, in the Solution Explorer, right-click the solution and select Add ➤ New Item. In the Add New Item dialog box, select Screens in the Installed Templates pane, select the screen type that you want to create, and then click Add. In the Select Business Object dialog box, click Search and in the Search for Business Object dialog box, specify the business object that you want your screen to be based on.

- **OWL Excel Export for Business Objects**
  Object Work Lists (OWL) generated for custom business objects allow you to export the OWL data to a Microsoft Excel file. The file is based on a standard template, which you cannot change. However, you can format the generated spreadsheet as required.
  For more information, see Create an OWL Excel Export [page 316].

**See Also**

What's New in the Syntax for Business Objects and Business Object Extensions [page 27]
What's New in the Syntax for Implementation of Business Logic [page 28]
What's New in Reuse Libraries and Built-In Functions [page 31]

2.8 What's New in Business Configuration

This document provides information about what is new in business configuration in the SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

**What Is New**

The following functions are new.
You can create business configuration content for a solution. In the Solution Explorer, you can access the following menu items by right-clicking the Business Configuration node.

- **Create BAC Elements**
  - You can create a business option when no selective adaptation of business configuration content is required.
    For more information, see Create a Business Option [page 281].
  - You can create a business topic when you need to provide multiple business options to key users.
    For more information, see Create a Business Topic and Business Options [page 283].
  - You can define a country if your solution requires country-specific scoping for a country that is not supported by the standard SAP Business ByDesign solution.
    For more information, see Create a Country and Business Options [page 286].

- **Create Business Configuration Set**
  You can create a business configuration set (BC set) to configure the behavior of business processes in your solution and define list data types that you can use to model fields in a business object. You can also create a BC set using an SAP business configuration object (BCO) to configure the behavior of standard business processes in the SAP solution.
  For more information, see:
  - Create a BC Set with an Implicit BCO [page 289]
You can only create a BC set using an SAP BCO if you are developing the solution on a development tenant. For information, see Lifecycle Management of Customer-Specific Solutions on a Development Tenant [page 139]. You cannot create BC sets using SAP BCOs in a solution template.

Create Business Configuration Object
You can create a custom business configuration object (BCO) to define solution-specific business configuration content on which you can base a BC set. For more information, see Create a Business Configuration Object [page 292].

Create Business Configuration View
You can create a business configuration view (BC view) to allow key users to change the values defined in a BC set during fine-tuning. For more information, see Create a Business Configuration View [page 295].

2.9 What’s New in Analytics

This document provides information about what is new in analytics in the SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

What Is New
The following functions are new:

- Access Control for Data Sources
  You can enable access restrictions for data in reports that are based on a data source you created in a solution. To enable access control, in the Data Source Wizard, in the Create a Data Source step, select Enable Access Control in the Basic Information section.
  The access restrictions of the anchor business object are inherited; therefore customers can only assign reports based on this data source to work center views with the same access control.
  For more information, see Create a Data Source [page 336].

- Referencing of a Data Source
  You can allow referencing of a data source you create in a solution by another data source and allow the fields of your data source to be used in reports based on the referencing data source. You can also specify a target user interface (UI). Users will then be able to navigate from a report based on this data source to this target UI. If you want to enable navigation from a report to another floorplan that is not a standard UI, you must define this UI.
  To allow referencing, in the Data Source Wizard, in the Create a Data Source step, select Allow Referencing in the Basic Information section.
  To specify a target UI for navigation, in the Data Source Wizard, in the Create a Data Source step, select Allow Referencing in the Basic Information section; then under Object Based Navigation, specify a target UI.
  For more information, see Create a Data Source [page 336].

What Has Changed
The following functions have changed:
• **Naming of the Steps in the Data Source Wizard**
  The titles of the steps in the *Data Source Wizard* have changed.

<table>
<thead>
<tr>
<th>Feature Pack 3.5 and Earlier</th>
<th>Feature Pack 4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter Header Information</td>
<td>Create a Data Source</td>
</tr>
<tr>
<td>Define Transformations</td>
<td>Define Transformation Rules</td>
</tr>
<tr>
<td>Define Aggregations</td>
<td>Define Aggregation Behavior</td>
</tr>
</tbody>
</table>

For information about the functions that are new and the functions that have changed in the SAP Business ByDesign key user tools for analytics see the What's New documentation for SAP Business ByDesign Feature Pack 4.0.

## 2.10 What's New in Extensibility

This document provides information about what is new in extensibility in the SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

### What Is New

The following functions are new:

- **Creation of Enhancement Implementations of SAP Enhancement Options**
  SAP provides enhancement options that are released with the public solution model. An enhancement option is a repository object for a specific application that allows the standard behavior of the application to be enhanced without modifying the standard SAP Business ByDesign solution. A distinction is made between single-use enhancement options and multiple-use enhancement options.
  You can create an enhancement implementation of an SAP enhancement option to allow customers to apply business rules that meet specific requirements, for example, country-specific requirements. If the enhancement option for which you create an implementation is for single use only, you must define a business configuration set (BC set) using an SAP business configuration object (BCO) so that you can enhance the filter with the value that you want to use in your filter definition.
  For more information, see [Enhancement Implementation Quick Guide](#).
  To create an enhancement implementation, in the *Solution Explorer*, click the *Add New Item* button; then in the *Add New Item* dialog box, select the *Enhancement Implementation* template.
  For more information, see [Create an Enhancement Implementation](#).

- **Documentation of SAP Enhancement Options**
  In the SAP Business ByDesign studio, you can explore SAP enhancement options and view the documentation of the enhancement options using the *Repository Explorer* tool window. The documentation of the enhancement options describes how you can use an enhancement option, when the enhancement option is called, and what it does. Furthermore, it gives information about the business configuration content you need to develop for an enhancement implementation, the scoping requirements in the customer system, as well as information about the filters, the input parameters and output parameters, and testing activities.
  You can open the *Repository Explorer* from the *View* menu by clicking [Other Windows > Repository Explorer](#). You can dock the tool window on the edge of the developer desktop or as a tabbed document. For best readability, we recommend that you dock the *Repository Explorer* as a tabbed document. For more information, see [Repository Explorer](#).

- **Examples of Use Cases for SAP Enhancement Options**
The documentation of the SAP Business ByDesign studio includes the following example of use cases for SAP enhancement options:

**Creation of Bank Statement File Formats**
You can create bank statement formats to meet the business requirements of customers located in or dealing with countries for which a localized version of the SAP Business ByDesign solution is not available. You can then map the a bank statement file to the structure of the Bank Statement Format business object. The customer can then update the system scoping and use this format when processing automatic bank statements.
For information about the end-to-end process, see Creating Bank Statement File Formats.

**What Has Changed**
The following function has changed:

- **Form Data Type Extensions No Longer Available**
  SAP no longer provides the following:
  - Creation of new form data type extensions
  - `GetEmptyLine` method for collections

  This means that you cannot create a new form data type extension definition or add the `GetEmptyLine` method to a script file anymore.
  If your solutions already contains a form data type extension definition or uses the `GetEmptyLine` method in a script file, your solution still works without errors in SAP Business ByDesign Feature Pack 4.0. However, a warning message is displayed when you activate the solution in the studio.

2.11 What's New in Mashups

This document provides information about what is new in mashups in the SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

**What Is New**
The following functions are new:

- **Web Service Authoring Renamed to Mashup Web Services**
  `Web Service Authoring` has been renamed to `Mashup Web Services`, since the Web services that you add here are used exclusively for mashups.

- **Use a Mashup Web Service in a Script File**
  If you have configured a REST or RSS/Atom Web service, you can implement the call of this Web service in a script file of your solution. For more information, see Using a Mashup Web Service in a Script File [page 396].

- **OData and SAPData Web Services No Longer Available**
  It is no longer possible to add OData and SAPData Web services for mashups.

- **Create Port Bindings**
  In previous releases, you could use port bindings provided by SAP only.
  Now you can create your own port bindings. This means that you can now specify other fields (including your own fields) as inports and outports of a mashup and are no longer tied down to using the port bindings provided by SAP. For more information, see Mashups Quick Guide [page 368].

- **Mashup on Its Own Tab Screen**
You can define a mashup to be displayed on the screen in a separate tab. This makes it possible for the mashup to write the data back into the system in its own area of a screen, making it easier for the user to view and work with the data provided by the mashup.

- **Support for Translation**
  In the Solution Explorer, when you right-click a mashup, a dropdown menu provides the following functions for translation:
  - Export Text for Translation
  - Import Translation
  - Check Translation Status

  This makes it possible for you to translate texts that are used in mashups that you have created. For information on translation of texts in the SDK, see Translation Quick Guide [page 263].

### What Has Changed

The following function has changed:

- **Add Mashups to Your Floorplans Without Extensibility Explorer**
  Adding mashups to floorplans that you have designed is now done without using the Extensibility Explorer and change transactions in the SAP UI Designer. This means that you are no longer limited to using SAP outports and inports that are set at predefined positions on the floorplan, and can add mashups to any part of the floorplan. For more information, see Add a Mashup to Your Floorplan [page 405].
  This function has been extended only to those floorplans that you have designed. The Extensibility Explorer is still used to add mashups to SAP floorplans.

- **Using a Mashup Web Service in a Script File**
  In previous releases, if you had configured a REST mashup Web service, you could implement the call of this Web service in a script file of your solution. You can now do this for RSS/Atom mashup Web services as well. For more information, see Using a Mashup Web Service in a Script File [page 396].

### 2.12 What's New in Web Services

This document provides information about what is new in Web services in the SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

### What Is New

The following functions are new:

- **External Web Service Integration**
  The External Web Service Integration Wizard enables you to integrate an external Web service into your solution. In this wizard, you can import the WSDL file of an external Web service. You can then address the Web service in your business logic in order to enable your solution to access a remote application using SOAP-based communication.
  For more information, see the Web Services Quick Guide [page 480].

- **Allow Nonsecure URLs (HTTP)**
  You can allow the usage of external http web services in an SAP Business ByDesign system. When you define the external web service integration scenario in the SAP Business ByDesign studio, you select the Allow Usag of Nonsecure URLs (HTTP) option. In SAP Business ByDesign, users are then able to define a communication arrangement using the http protocol.
For more information, see Allow Nonsecure URLs (HTTP) [page 500].

- **Communication Scenario Definition**
  The Communication Scenario Definition Wizard enables you to define a communication scenario. In this wizard, you can define a set of inbound and outbound services for a business process. Based on a communication scenario, a customer can create a communication arrangement in the SAP solution in order to exchange business documents electronically.
  For more information, see the Web Services Quick Guide [page 480].

- **Deprecation of A2X Web Service Operations**
  A2X Web service operations that SAP sets to deprecated in a release, are removed from the solution in the subsequent release.
  You can search for deprecated A2X Web service operations in the SAP solution in the Service Explorer view of the Application and User Management work center.
  The Operation Release Status field in the Service Explorer view specifies if a Web service operation is released or deprecated and the Operation Depreciation Release field specifies the release in which a Web service operation has been set to deprecated.
  Also in the Service Explorer view, the documentation for Web service operations is available. This documentation explains how to handle Web service operations that have been set to deprecated.
  For more information, see the Web Services Quick Guide [page 480].

### What Has Changed

The following function has been changed:

**Creation of Web Service Authorization**

Apart from the Web service authorization object, there is now an alternative method for authorizing users to call a Web service that you have created based on a business object. The customer can authorize users using a communication arrangement. Based on a communication scenario that you have created in the SDK, the customer can create a communication arrangement in the SAP solution. In the communication arrangement, the customer can provide an authentication method to authorize a user to call the Web service.

#### 2.13 What's New in the Syntax for Business Objects and Business Object Extensions

This document provides information about what is new in business objects and business object extensions in the SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

### What Is New

The following functions are new:

**Business Objects**

- **Default Values**
  In previous releases, you could define default values only via the Add Field function in the key user tools in the SAP solution. As of Feature Pack 4.0, you can use the SDK to define default values for business objects.
  For more information, see Default Values (Business Object) [page 162].

- **Labels and Tooltips**
  You can now define label and tooltip texts for fields in the business objects you create using the Label and Tooltip annotations.
For more information, see Label (Business Object) [page 165] and Tooltip (Business Object) [page 168].

- **Transient**
  You can use the Transient annotation to create transient fields in business objects.
  For more information, see Transient (Business Object) [page 168].

- **Valuation**
  The Valuation keyword enables you to create not only 0..1 and 1..1 associations but also 0..n and 1..n associations. Thus, you can model hierarchical business object structures, for example, a product catalog.
  For more information, see Valuation (Business Object) [page 169].

**Business Object Extensions**

- **Default Values**
  In previous releases, you could define default values only via the Add Field function in the key user tools in the SAP solution. As of Feature Pack 4.0, you can use the SDK to define default values for business object extensions.
  For more information, see Default Values (Business Object Extension) [page 172].

- **Reference (Business Object Extension)**
  The Reference annotation enables you to use an element declared in one business object extension in another business object extension.
  For more information, see Reference (Business Object Extension) [page 176].

- **Relation (Business Object Extension)**
  The Relation annotation enables you to create a reference to a business object by using the identifier of an element as the target of the reference.
  For more information, see Relation (Business Object Extension) [page 177].

### 2.14 What's New in the Syntax for Implementation of Business Logic

This document provides information about what is new in the syntax for the implementation of business logic in SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

**What Is New**

The following functions are new:

**Code Editor**

- **F1 Help for Keywords**
  You can access context-sensitive help for keywords when you implement the business logic for your solution in script files. To do this, in the code editor, place your mouse cursor on a keyword and press F1. For more information, see What’s New in the Integrated Development Environment [page 16] under Context-Sensitive Help (F1) for Keywords (Business Logic).

- **Personalizing Font Style, Font Size, and Font Color for Elements in the Editor**
  You can now use the Administration Options and Settings menu function to change the font style, the font size, and the background and foreground font color for keywords, comments, identifiers and literals in the code editor for script files. To access these settings, in the Options dialog box select Fonts and Colors in the Environment folder. In Show settings for, select Text Editor and change the font style, font size, and font color options for code elements that are related to the business logic, for example, Business Logic Comment (SAP) or Business Logic Keyword (SAP).
  The settings are stored on your local PC and will take effect immediately after you have saved them.
• **Icons in Code Completion**
  The code completion now displays a number of new icons which provide more information about the relevant element or entity. The icons can, for example, reflect the multiplicity of an element or association or the type of an association and they indicate whether an element or entity is read-only or deprecated. For more information about the deprecation mechanism for SAP entities, see *About the Public Solution Model (PSM)* [page 66] under *Understanding Deprecation*.

![Example of the code completion displaying icons](image)

• **Detailed Descriptions in Code Completion**
  The descriptions in the code completion now provide more detailed information about the corresponding element.
  - **Information about the Target of an Association**
    The description of an association provides detailed information about the target of the association.

  ```javascript
  var itemProduct : SalesOrder.Item.ItemProduct;
  var : = itemProduct;
  ```

  - **Enhanced Type Information About Elements**
    The descriptions provide detailed type information about an element, for example, the base type, the length, the alpha conversion flag, or whether the type is a collection.

  ```javascript
  var itemProduct : SalesOrder.Item.ItemProduct;
  itemProduct.Material.VL
  ```

  - **Enhanced Type Information about Generic Methods**
    The type information for generic methods, such as *Add*, *Create*, or `<query>.Execute`, now also provides the actual type of the collection, business object or query.

  ```javascript
  query.Execute()
  ```

**What Has Changed**

The following functions have changed:
Code Editor

- **Code Completion**
  The code completion for keywords is no longer context-sensitive. That is, the list of the code completion contains any available keyword, even if it is not valid in the current context. The code completion for identifiers, however, still offers only valid options.

Syntax

- **Clear Method**
  You can now use the `Clear` method for variables defined with the `elementsof` modifier.
  For more information, see **Clear Collection** [page 184].

- **OrderBy, OrderByDescending and DistinctBy Method**
  These methods now also support collections of data types.
  For more information, see **Sort Collection** [page 200] and **Remove Duplicate Rows from Collection** [page 197].

- **Variable Type in Variable Declarations**
  Usually, the variable type is automatically inferred upon first assignment of a variable. Now you can also explicitly specify the variable type. For this, you can use basic data types, global data types (GDTs) and elements of an aggregated data type. You can, for example, specify the element of an aggregated data type in an enhancement implementation to create the return value.
  You can also determine the type of a variable by referencing a signature element of a library function. This is helpful if you prepare the request data to call a Web service client library.

- **Variable Declarations for Collections**
  You can now use both the `collectionsof` modifier and the `elementsof` modifier to define variables for collections.

```plaintext
import ABSL;
import AP.Common.GDT;
import AP.CustomerInvoicing.Global
var values : collectionof Amount;
var CirData : collectionof elementsof CustomerInvoiceRequest;
```

For more information, see **Declarations** [page 188].

- **Precedence Rules for Custom Entity Names**
  The existing name qualification rule that custom entities have precedence over SAP entities now applies to additional entities. Up to now this rule applied to the top-level entities business objects, data types and libraries. For example, if you use an `Amount` business object that you have defined and the `Amount` GDT provided by SAP in the same script file, you need to fully qualify the GDT.
  This rule now also applies to sub-elements, for example, actions and node elements.
  For more information, see **Name Qualification** [page 193].

- **Calling Web Services**
  If you have integrated an external Web service into your solution, you typically call the service operations provided by the Web service in a script file. The external Web service integration generates a library in the project namespace. This library is similar to a reuse library and you can use it to implement your business logic to call the service operations.
  For more information, see **Define the Business Logic to Call a Service Operation** [page 505] and **Sample Code: Event BeforeSave Web Service Operation** [page 506].
2.15 What's New in Reuse Libraries and Built-In Functions

This document provides information about what is new in the reuse libraries and built-in functions of the SAP Business ByDesign studio Feature Pack 4.0 for the development of customer-specific solutions.

What Is New

The following functions are new:

- **Basic Reuse Libraries**
  The following library and service are new:
  - Library Binary
    You can use the service of this reuse library to convert a string to a binary value. For information, see Binary [page 203].
  - Library Context, Service GetSystemURL
    You can use this new service of the Context reuse library to return the URL of the system. For information, see Context [page 204].

- **Reuse Libraries for Business Areas**
  The following reuse libraries for business areas are new:
  - Date Time Utilities
    You can use the service of this reuse library to determine the next working day. For more information, see Date Time Utilities [page 225].
  - Output Management Utilities
    You can use the service of this reuse library to provide business object data in a portable document format (PDF) document. For more information, see Output Management Utilities [page 227].

- **Built-In Functions for Strings**
  The following built-in functions for strings are new:
  - FindLast
    Searches from right to left and returns the position of a substring in a string.
  - FindRegEx
    Searches from left to right and returns the position of a regular expression pattern in a string
  - FindLastRegEx
    Searches from right to left and returns the position of a regular expression pattern in a string
  - Trim
    Deletes leading and trailing characters.
  - TrimLeft
    Deletes leading characters.
  - TrimRight
    Deletes trailing characters.
  - ReplaceRegEx
    Returns all locations in a string with a substring that matches a regular expression pattern.

For more information, see Built-In Functions for Strings [page 230].
What Has Changed

The following functions have changed:

- **Generic ToString Built-In Function**
  The generic `ToString` built-in function now also converts binary values. For more information, see *Generic Built-In Functions* [page 229].

- **Find Built-In Function for Strings**
  If you use the `Find` built-in function, you can now optionally define a search area by specifying the start position only or by specifying the start position as well as the length of the search area.

```csharp
this.string = "aaabbbccccbbb";
this.result = this.string.Find("bbb", 5, 3);
```

2.16 What's New in the Public Solution Model

This document provides information about what is new in the public solution model (PSM) in the SAP Business ByDesign studio Feature Pack 4.0 for customer-specific solutions.

What Has Changed

The following has changed:

- **SAP Business Objects**
  The restriction that SAP business objects are read-only if they are used or referred to in customer-specific solutions is still valid. Therefore, you cannot create, update, or delete node instances of SAP business objects or modify them, for example, by using associations. However, you can now request write-access to SAP business objects that are released with the PSM. To do this, you can report an incident on the customer’s test tenant.
  Before you upload the customer specific-solution to the customer’s productive tenant, report another incident on the customer’s productive tenant to request write access for the same SAP business objects in the productive tenant.
  For information about how to report an incident, see [here] [page 262].
3 Getting Started

3.1 Getting Started with the SDK

Getting Started encompasses the following reference material:

<table>
<thead>
<tr>
<th>First Steps</th>
<th>Introducing the SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learn about the following:</td>
<td>Learn about the following:</td>
</tr>
<tr>
<td>• Installing the Software Development Kit [page 37]</td>
<td>• Features of the SDK</td>
</tr>
<tr>
<td>You have to install the latest version of the software development kit (SDK) from the SAP Service Marketplace. This document takes you through the requirements and steps to install the SDK</td>
<td>The SDK enables you to adapt and enhance the solution capabilities of the SAP solution. Using the SDK, you can develop, deploy, and test specific add-on functionality. For an overview of the SDK features, see About the Software Development Kit (SDK) [page 35].</td>
</tr>
<tr>
<td>IMPORTANT: Prior to working in the SDK, you must change the initial password provided to you in the SAP solution.</td>
<td>• Working in the Integrated Development Environment [page 44]</td>
</tr>
<tr>
<td>• User Setup Quick Guide</td>
<td>The development tools provided by the SDK are presented in an integrated development environment (IDE) based on Microsoft Visual Studio. The user interface (UI) of the SDK is referred to as the developer desktop.</td>
</tr>
<tr>
<td>Before you can begin working in the SDK, the initial user has to create the required users for the SDK and for the SAP solution. This quick guide explains the entire process from creating service agents, requesting users for these service agents, assigning an ID and an initial password to each user, and assigning the required work centers. In addition, it also explains the process to create business users for testing purposes in the SAP solution. For more information, see User Setup Quick Guide [page 39].</td>
<td>• Working with Wizards [page 65]</td>
</tr>
<tr>
<td>• User Roles [page 38]</td>
<td>The wizards guide you through a logical sequence of steps, making it easier and quicker for you to perform activities that are complex, unfamiliar, or infrequently performed. All wizards are easy to access and follow similar user interface patterns</td>
</tr>
<tr>
<td>Find out about the different user roles in the SDK:</td>
<td></td>
</tr>
<tr>
<td>○ For development purposes: SDK Administrator and Developer</td>
<td></td>
</tr>
<tr>
<td>○ For testing purposes: Business User</td>
<td></td>
</tr>
</tbody>
</table>
Get an Introduction to Business Object Modeling and the Public Solution Model

- **Introduction to Business Object Modeling**  [page 70]
  The SAP solution is built using the concept of business objects to model the business environment.

- **Public Solution Model**
  The public solution model contains the released content that can be used in the SDK. The information about the released content types is available in the Business Center at [https://www.sme.sap.com](https://www.sme.sap.com) under **SAP Business ByDesign** » **Community** » **Wiki Info Exchange** » **SAP Solutions On Demand Studio** in the section **Partner/Developer Information** → **Public Solution Model**.

Here you can find information about released business objects, embedded components, object value selectors, UI imports, UI exports, and data types.

- **Test Shell Quick Guide**  [page 85]
  Learn how the **Test Shell** work center view in the SAP solution enables you to browse the SAP business objects and entities that you can use in developing your solution.

  **NOTE:** The **Test Shell** functionality is only available to partners in their partner development tenants. The functionality is not available in customer systems or customer-specific development systems.

---

Learn How to Build Solutions

- **Example: Building a Solution**  [page 88]
  In this example you build a solution for the SAP solution to manage the allocation of car parking spaces to employees.

- **Enablement Service**
  SAP provides an **SAP Business ByDesign Studio** Experience enablement service. Further details related to curriculum, prerequisites, pricing, delivery format, schedule, and registration can be found in the Business Center at [https://www.sme.sap.com](https://www.sme.sap.com) under **SAP Business ByDesign** » **Learn** » **Enablement Services** → **ByD Studio Experience**.

---

Check Out the Development and Mobile Development Guidelines

Check out the guidelines in the Business Center at [https://www.sme.sap.com](https://www.sme.sap.com) under **SAP Business ByDesign** » **Community** » **Wiki Info Exchange** » **SAP Solutions On Demand Studio** in the section **Partner/Developer Information** under **Topics in Detail** → **Documentation**:

- **Development Guideline**
  The Development Guideline provides general programming rules and guidance on selected product standards.

- **Mobile Development Guideline**
  The Mobile Development Guideline provides general UI guidelines related to mobile development.

Find Out How You Can Localize Your Solution

Check out the **Localization Toolkit**  [page 125]

The localization toolkit is a set of development tools and instructions that enable SAP partners to extend and adapt the capabilities of the SAP Business ByDesign solution to provide a locally-compliant solution for customers in countries where a fully-localized SAP country version is not available. You can access the toolkit in the Business Center at [https://www.sme.sap.com](https://www.sme.sap.com) under **SAP Business ByDesign** » **Community** » **Wiki Info Exchange** » **Country Information** » **Customer-Specific Localization** → **Localization Toolkit**.

---

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages caused by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.
3.2 About the Software Development Kit (SDK)

The software development kit (SDK) enables SAP partners, resellers, and customers to adapt and enhance the solution capabilities of SAP Business ByDesign. The development tools provided are presented in an integrated development environment (IDE) based on Microsoft Visual Studio. Using the SDK, you can develop, deploy, and test specific add-on functionality. Furthermore, you can create and integrate new business content, services, and user interfaces to provide complete micro-vertical business solutions.

The SDK allows SAP partners, resellers, and customers to work with the same entities that SAP uses in the core development of its solution and to develop solution capabilities that have the same look and feel as the standard SAP solution.

Solutions and Solution Templates

Customer-Specific Solutions

The development of a customer-specific solution is triggered by a customer requirement. The solution is typically developed by a reseller for a specific customer.

For information about the development scenarios for customer-specific solutions, see Lifecycle Management of Customer-Specific Solutions [page 127].

Solution Templates

With a solution template you can organize common development content that you want to reuse for customer-specific solutions. Solution templates enable you to rapidly start the development of customer-specific solutions, for example, for a specific industry.

For more information, see Lifecycle Solution Templates Quick Guide [page 149].

Features

Integrated Development Environment

The SDK is based on a local integrated development environment (IDE) that:

- Provides access to a hosted repository.
- Provides access to all tools you need to create and enhance the functionality of the standard SAP solution.
- Is integrated with the tools that key users work with in the SAP solution.
- Allows you to manage the entire lifecycle of customer-specific solutions, including development, testing, and assembly.
- Makes it easy and quick for you to perform activities through features such as wizards for multi-step tasks.
- Simplifies programming through features such as code completion.

For information about the specific features and tools provided by the SDK, see Working in the Integrated Development Environment [page 44].
Lightweight Scripting Language
You use the scripting language to define business objects and business object extensions and to implement the business logic for solution capabilities. The scripting language is a lightweight scripting language. Its syntax is easy to learn and to enter using a simple code editor.
For information, see Scripting Language Reference [page 156].

Model-Based Development
The SAP public solution model (PSM) contains all entities in the standard SAP solution that SAP has released for use by external consumers. External consumers can be partners who develop solutions such as add-ons and integration scenarios on top of the SAP solution or key users who, for example, use SAP data sources to create new reports.
For information about the basic principles of the PSM and SAP content types, see About the Public Solution Model (PSM) [page 66].

UI Designer for Desktop and Mobile User Interfaces
You use the UI designer to create new and replace existing user interfaces (UI). You can create and modify UI components using floorplans, patterns, and UI controls.
For information about the UI designer, see Introduction to the User Interface Designer [page 508].

Integration with Services
You can generate web services to access SAP business objects and the business objects of add-on solutions. You can also set up communication between business objects residing in different deployment units and import external XML data into business objects.
For information about web services, see Web Services Quick Guide [page 480].
You can also integrate data from the SAP solution with data provided by an online web service or application by creating mashups. Mashups can include web searches, company or industry-specific business information or online map searches.
For information about mashups, see Mashups Quick Guide [page 368].

Translation Support
You can extract text from certain project items and export it in an XML-based file format for translation into one of the languages supported by the standard SAP solution.
For information about the functions that support translation of a solution, see Translation Quick Guide [page 263].

End-to-End Software Lifecycle Management
Integration with the SAP solution provides a strong lifecycle management backbone that supports the full lifecycle of solution. For information about the development scenarios for customer-specific solutions, see Lifecycle Management of Customer-Specific Solutions [page 127]

On-Demand Readiness
The SDK supports the development of solution capabilities that are tailored for specific customers. The development and deployment of these solution capabilities causes no disruption to daily business because the upload process of a solution to a customer’s production system does not require any system downtime.
3.3 First Steps

3.3.1 Installing the Software Development Kit

Overview
If you are building solutions for SAP Business ByDesign, you install the SAP Business ByDesign studio.

Prerequisites

- You are using one of the following operating systems:
  - Microsoft Windows XP operating system
  - Microsoft Windows Vista
  - Microsoft Windows 7 operating system
- You have installed the following:
  - Relevant version of Microsoft .NET Framework
  - Relevant version of Microsoft Visual Studio 2010 Shell (Isolated)
  - Relevant version of Microsoft Silverlight

Partners are required and solely responsible to regularly install security updates or patches provided by Microsoft Corporation for the Microsoft .NET Framework and Microsoft Visual Studio 2010 Shell (Isolated). As these components need to be installed/obtained by the partner, SAP is not responsible for installation of any related security updates or patches. For more information, see [http://www.microsoft.com](http://www.microsoft.com).

- The build version of the SDK must match the release version of the SAP solution for which you are creating a solution.
  To determine the release version, in the SAP solution, click `Help > About ...` and verify the `Technical Release` and `Changelist` details match the latest build version of the SDK.
- Before installing the latest build, you must remove the previous version.
- You have an S-User for the SAP Service Marketplace.
  If you do not have access, contact Partner Management.

Steps

2. Click SAP Support Portal, and select the `Software Downloads` tab.
3. In the SAP Software Download Center, click Support Packages and Patches.
5. In the Search Term field enter **ByD studio** and click **Search**. The Search Results appear.

If you know which build you require, you can also navigate directly to the correct release. You can find details in the Business Center at the Installation Corner (quick link: [https://wiki.sme.sap.com/wiki/x/wwVNCw](https://wiki.sme.sap.com/wiki/x/wwVNCw)). For more information, see also SAP Note 1567861 at [https://service.sap.com/sap/support/notes/1567861](https://service.sap.com/sap/support/notes/1567861).

6. Download the relevant version of the SDK. Here you can also find more information about the installation of the SDK and the requirements for specific releases.

### 3.3.2 User Roles

The SDK has the following user roles:

**Developer**

Developers can perform the following tasks:

- Create and maintain all types of development objects, such as business objects and user interfaces, in the SDK, user interface (UI) designer, and the tools in the SAP solution
- Access work center views for Analytics and Forms

For more information about what the developer role can do, see the reference documentation in the section [Developer Desktop](#) [page 260].

**SDK Administrator**

SDK administrators can perform all tasks that a developer can perform. In addition, SDK administrators can perform the following tasks:

- Create new solutions
- Delete solutions
- Delete sessions and locks for other users

For more information about the tasks that only an SDK administrator can perform, see:

- Administration Quick Guide for Customer-Specific Solutions [page 144]
- Implementation Manager Quick Guide for Customer-Specific Solutions [page 146]

**Business User**

The business user role is used for testing within the development or test tenant of tools in the SAP solution. Business users have authorizations and restrictions equivalent to users on a customer tenant; they have no authorization within the SDK.

Business Users can perform the following tasks (depending on their access rights):

- Create additional business users
- Assign work center views to other business users
- Grant instance-based authorizations to other business users
- Perform business configuration scoping and deployment
- Start a trace from within the SAP solution

**Setting Up the Users**

For information on how to set up the users for each of the roles, see User Setup Quick Guide [page 39].

**3.3.3 User Setup Quick Guide**

Before you can begin administrating, developing, or testing solutions in the software development kit (SDK), you have to create development users (SDK administration and developer user roles) and business users. You create these users in SAP Business ByDesign.

After these users have been created, the development users must change their initial passwords and maintain contact data. Business users only need to change their initial passwords.

⚠️ Development users with SDK administration or developer user roles are for use in the SDK only.
Business users are for use in the SAP Business ByDesign solution.

**Business and Technical Background**

**User Roles**

The user roles are SDK Administrator, Developer, and Business User.

For more information, see User Roles [page 38].

**Administration Toolbar**

The SDK administrator user has authorization for the Administration toolbar, which provides access to a number of administrative tasks, such as creating a new solution.

For more information, see:
User and Access Management in SAP Business ByDesign

The User and Access Management view/tab allows you to manage users and assign access rights.

For more information, see:
- Business Users Quick Guide
- User and Access Management
- Work Quick Guide (in User and Access Management)

For information about logging on to SAP Business ByDesign, see Logging On to the System.

Tasks

Prerequisites

- Prerequisites for creating service agents, assigning user IDs and passwords, assigning work centers/tabs, and creating business users:
  In SAP Business ByDesign:
  - You have an initial business user (ADMINISTRATION01). The initial user is provided to you during the tenant provisioning process. Only use the initial business user in SAP Business ByDesign. Do not use it in the SDK.
  - You have access rights for the Business Partner Data work center and the User and Access Management work center.
- Prerequisites for maintaining contact data:
  You have a service agent user ID, which matches the technical ID of the service agent.

Tasks of the Initial User

Before starting to develop in the SDK, you have to create the required development users with the relevant administration or developer rights. You do this by creating service agents in SAP Business ByDesign, and assigning the relevant user ID, password, and work centers to those service agents.

Create a Service Agent and Request a User (for Development Users)

1. In the Business Partner Data work center, open the Service Agents view.
2. Click [New] and select Service Agent.
The New Service Agent quick activity opens.
3. Enter a service agent ID and a name.
   If the Service Agent ID field is enabled, your entry must have the prefix EXT.
   If internal number assignment is active, the system determines the new ID.
   For the ID, the system uses the naming convention <LastName><FirstName><ServiceAgentID>.
   For example, the service agent ID for Kate Jacob could be JACOBKATEEXT999.
4. Note down the service agent ID.
5. Save the service agent and click [Request User].
The system creates a user for the service agent.
6. Click [Save] and close the editor.
Assign a User ID and a Password to the Service Agent (for Development Users)

1. In the Application and User Management work center under the User and Access Management view, open the Business Users subview.
2. Search for the service agent (business user) that you created. For the search, enter one of the following:
   - *<ServiceAgentID> (for example, *EXT999)
   - Name
3. Click Edit and select Attributes. The Business User quick activity opens.
4. In the User Data screen area, define a user ID, for example, the last name of the service agent.
5. In the Language field, set the language to English. You must set the language to English to ensure consistency of your solution, because the logon language of both SAP Business ByDesign and the SDK is English.
6. In the fields Password and Confirm Password, assign an initial password.
7. Click Save or Save and Close.

Assign Work Centers to the Service Agent (for Development Users)

1. In the Application and User Management work center under the User and Access Management view, open the Business Users subview.
2. Search for the service agent (business user) that you created. For the search, enter one of the following:
   - *<ServiceAgentID> (for example, *EXT999)
   - Name
4. In the Edit Access Rights editor, on the Work Center and View Assignment tab assign the following work centers and views by selecting Assigned to User.

<table>
<thead>
<tr>
<th>Work Center ID / Work Center Name (in Available Work Centers)</th>
<th>Work Center View ID / Name (in Available Views in Work Center: &lt;Name of Work Center&gt;)</th>
</tr>
</thead>
</table>
| PDI_PARTNER_DEVELOPMENT / Partner Development                | • Developer User Role  
  Select the following views:  
  ○ PDI_DEVELOPMENT / Development  
  ○ PDI_TS_BO_MD / Business Objects  
  ○ PDI_TS_EC_MD / Embedded Components  
  ○ PDI_TS_OVS_MD / Object Value Selectors  
  • Administrator User Role  
  Select all of the following views:  
  ○ PDI_ADMINISTRATION / Administration  
  ○ PDI_DEVELOPMENT / Development  
  ○ PDI_TS_BO_MD / Business Objects  
  ○ PDI_TS_EC_MD / Embedded Components  
  ○ PDI_TS_OVS_MD / Object Value Selectors |

You need at least one user with an administrator role. For more information, see User Roles [page 38]
<table>
<thead>
<tr>
<th>Work Center ID / Work Center Name (in Available Work Centers)</th>
<th>Work Center View ID / Name (in Available Views in Work Center: &lt;Name of Work Center&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANA_BUSINESSANALYTICS / Business Analytics</td>
<td>Select all of the following views:</td>
</tr>
<tr>
<td></td>
<td>• ANA_DESIGNREPORTS / Design Reports</td>
</tr>
<tr>
<td></td>
<td>• ANA_DESIGN_DSO / Design Data Sources</td>
</tr>
<tr>
<td>ITS_APPLICATIONUSERMANAGEMENT / Application and User Management</td>
<td>Select all of the following views:</td>
</tr>
<tr>
<td></td>
<td>• ITS_FORMTEMPLATEMAINT / Form Template Maintenance</td>
</tr>
<tr>
<td></td>
<td>• ITS_MASTERTEMPLATEMAINT / Master Template Maintenance</td>
</tr>
<tr>
<td></td>
<td>• ITS_OFFICETEMPLATEMAINT / Office Template Maintenance</td>
</tr>
<tr>
<td></td>
<td>• ITS_KEYUSERCHANGELOG / Flexibility Change Log</td>
</tr>
<tr>
<td></td>
<td>• MAS_MASHUPAUTHORING / Mashup Authoring</td>
</tr>
<tr>
<td></td>
<td>• MAS_WEBSERVICEAUTHORING / Web Service Authoring</td>
</tr>
<tr>
<td></td>
<td>• ITS_INCIDENTS</td>
</tr>
</tbody>
</table>

5. Click [Save] or [Save and Close].

6. Notify the users that you have created a user ID and an initial password.

The users must change the initial password. To do this, they must log on for the first time to SAP Business ByDesign using the assigned user ID. The system requests a new password. The user can use the same password in both SAP Business ByDesign and in the SDK.

In addition to the users for the SDK, you also have to create business users for use in SAP Business ByDesign only. These business users are for test purposes and are not for use in the SDK.

Create and Assign Business Users (Test Users)

The Business User role is used for testing within development tenant or test tenant of SAP Business ByDesign. Business users have authorizations and restrictions equivalent to users on a customer tenant.

Do not assign the Partner Development work center and its views to a business user role. These users must not have access to the SDK.

You use business users to perform the following in SAP Business ByDesign:

- Scoping and deployment of your business configuration entities
- Testing of your solution
- Starting a trace from within SAP Business ByDesign
Steps

1. Create the required business users (key users and end users) in SAP Business ByDesign.
2. Assign the required work centers and work center views to the business users.

To enable testing of business configuration entities, at least one standard business user must be assigned to the Business Configuration work center.

Do not test or change business configuration entities with a development user.

3. Grant instance-based authorizations to the business users.

See Also:

- Business Users Quick Guide
- Tracing Quick Guide [page 424]

Development Users

Change Your Initial Password

After receiving your user details, before logging on to the SDK, you must first change your initial password in SAP Business ByDesign. This is important, because you cannot change the initial password in the SDK.

1. Launch SAP Business ByDesign.
2. Enter your user ID and the initial password that has been created for you by the Administration01 user. The system requests a new password.
3. Enter a new password and confirm the entry.
   You can now also log on to the SDK with your user ID and the new password.

Do not change the logon language defined for your user ID, as this causes inconsistencies in the solution. The logon language must be English.

Maintain Your Contact Details

As a development user, you may have to create incidents for SAP Support. To ensure that these incidents are sent directly to SAP Support, you must maintain your telephone number and e-mail address.

For information about reporting an incident in SAP Business ByDesign, see Report an Incident [page 262].

If you do not maintain your contact details, when you create an incident the system forwards the incident automatically to a development user in your organization, who has maintained contact data.

1. Log on to SAP Business ByDesign with your development user.
2. Go to the Home work center and navigate to Self-Services Self-Service Overview Edit My Contact Data.
3. Enter your telephone number and e-mail address and save your entries.

Business Users

Change Your Initial Password

After receiving your user details, you must first change your initial password in SAP Business ByDesign.

1. Launch SAP Business ByDesign.
2. Enter your user ID and the initial password that has been created for you by the Administration01 user. The system requests a new password.
3. Enter a new password and confirm the entry.

3.4 Introduction to the Integrated Development Environment

3.4.1 Working in the Integrated Development Environment

The development tools provided by the SDK are presented in an integrated development environment (IDE) based on Microsoft Visual Studio. The user interface (UI) of the SDK, referred to as the developer desktop, is composed of different elements. Many elements are standard features and tools of Microsoft Visual Studio, for example, the Menu toolbar, the Standard toolbar, various tool windows, and the document window. In addition, the developer desktop provides UI elements that represent specific features and tools of the SDK development environment. The UI elements of the developer desktop are docked or auto-hidden on the left, bottom, and right. The tool windows, menus, and toolbars available depend on the type of solution or item you are working on as well as on the settings you have applied. You can move and position the UI elements to suit your working behavior.

Product Documentation

- You can access the SDK documentation from the Help menu by selecting SAP Business ByDesign Studio Help or SAP Cloud Developer Studio Help. The library documentation appears in your Web browser.
- You can install the latest SDK documentation in the integrated development environment (IDE) and view this Help content using Microsoft Help Viewer. The Help Viewer is installed when you install the Microsoft Visual Studio 2010 Shell (Isolated) Redistributable Package. You need to install the Help content manually. For more information, see Help Content Installation [page 47].

This documentation does not describe the standard features and technologies in Microsoft Visual Studio. For information about Microsoft Visual Studio, see http://www.microsoft.com.

Developer Desktop

The following sections describe the tools that you can use for the administration and lifecycle management of a solution and how you can access the corresponding user interface (UI) elements.

Administration toolbar

The Administration toolbar provides functions that allow you to create and delete a solution and create a patch. You can also access the Session Administrator tool window from the Administration toolbar.

You can access the Administration toolbar in the View menu by clicking Toolbars Administration. For more information, see Administration Quick Guide [page 144].

Session Administrator

The Session Administrator tool window displays a list of users in the SDK with open sessions. If you are assigned the Administrator role, you can delete user sessions and any item locks that are applied for the corresponding sessions.
All other users can delete their own sessions and item locks, but they cannot delete other users’ sessions and item locks.

You can access the Session Administrator tool window in the View menu by clicking Other Windows Session Administrator. You can also access the Session Administrator tool window from the Administration toolbar.

For information, see Administration Quick Guide [page 144].

Admin Log

If you encounter any problems while deploying your solution, you can check whether any errors have occurred by opening the Admin Log tool window. You can access the Admin Log tool window from the Administration toolbar.

For more information, see Administration Quick Guide [page 144].

Implementation Manager

The Implementation Manager is a tool for managing the lifecycle of customer-specific solutions. You can access the lifecycle management functions provided by the SDK from a toolbar and a tool window:

- **Implementation Manager** tool window
  The tool window provides a view of the solution status and implementation log at each stage in the process.
  You can access the Implementation Manager tool window in the View menu by clicking Other Windows Implementation Manager.

- **Implementation Manager** toolbar
  The toolbar provides buttons that allow you to assemble, upload, and create a patch for your solution.
  You can access the Implementation Manager toolbar in the View menu by clicking Toolbars Implementation Manager.

For more information, see Implementation Manager Quick Guide [page 146].

Administration menu

You can access existing administration and lifecycle management functions from the Administration menu. The Administration menu also gives you access to most of the functions provided by the My Solutions tool window.

The functions available on the Administration menu are:

- **Log On**
  For information, see Log On to the Repository [page 261].
- **Log Off**
- **Create Solution**
  For information, see Create a Customer-Specific Solution [page 152].
- **Delete Solution**
- **Session Administrator**
  For information, see Administration Quick Guide [page 144].
- **Implementation Manager**
  For information, see Implementation Manager Quick Guide [page 146].
- **Admin Log**
  For information, see Administration Quick Guide [page 144].
- **Switch Customer...**
  For information, see Switch a Customer Assignment [page 143].
- **Options and Settings**

The following sections describe the tools that you can use to develop a solution and how you can access the corresponding user interface (UI) elements.
My Solutions

The **My Solutions** tool window provides functions that allow you to create solutions and to access the repository. You must log on to a repository system to create and edit solutions in the SDK.

For more information, see [My Solutions](#page_53).

Repository Explorer

The **Repository Explorer** tool window displays SAP content that is released with the public solution model (PSM) such as business objects, data types, enhancement options, and inbound service interfaces. You can explore the content to gain an understanding of the repository structure and the business context and use of SAP content, which you can reuse and enhance in your solution.

For more information, see [Repository Explorer](#page_53).

Solution Explorer

The **Solution Explorer** tool window displays a solution and the project items created for this solution, for example, a business object, a screen, or a form. In the **Solution Explorer**, you can open items for editing and add new items to a solution.

For more information, see [Solution Explorer](#page_56).

Add New Item dialog box

The **Add New Item** dialog box displays the installed item templates that SAP provides for solution development. You can add items such as business objects, forms, different user interface (UI) components, and Web services to your solution.

For more information, see [Add New Item Dialog Box](#page_58).

Properties window

The **Properties** window displays information about the item in your solution that is currently selected in the **Solution Explorer**. When you select the solution, you can view solution properties such as the solution status. When you select an item such as a business object or a UI component, you can view item properties such as the activation status of runtime objects.

For more information, see [Properties Window](#page_63).

Options dialog box

The **Options** dialog box provides settings that allow you to configure the SDK to meet your needs. You can view or define information about the build, connectivity, debugging, runtime metadata, external tools, and tracing.

You can access the settings for the SDK in the **Tools** menu by clicking **Options**. In the navigation pane of the **Options** dialog box, click **SAP**. The options for the SDK are displayed on the right.

You can also access the settings for the code editor from the **Options** dialog box and define the default behavior of the code editor for business objects, business object extensions, form data type extensions, and actions and events. For example, you can display line numbers in the code editor for reference purposes. To access these settings, in the navigation pane of the **Options** dialog box, expand the **Text Editor** node and then expand the subnode for your project item.

You can access the connectivity settings where you can define repository system connections from the **Administration** menu by selecting **Options and Settings**.
Document windows

In the SDK, document windows provide a view of the items that you add to your solution. Examples of items that you can open, edit, and save in a document window are business object definitions, extensions for existing business object definitions, data sources, business object queries, and form definitions. The type of document window in which an item opens depends on the type of item. For example, when you open a business object definition, the business object file is displayed in the code editor.

Document windows are displayed on tabs in the center of the developer desktop. You can switch between open document windows by using the tabs. You can also view multiple document windows side-by-side.

You can open an item in a document window by double-clicking the item in the Solution Explorer.

If you double-click an item that represents a user interface component, such as a quick activity floorplan (QAF) screen or a business configuration view, the item file opens in a separate window in the UI designer.

Error List window

The Error List displays errors, warnings, and messages produced as you edit code, such as syntax errors.

You can display the Error List in the View menu by selecting Error List and then use the Errors, Warnings, and Messages buttons to select the entries you want to display. By default, the Error List is displayed at the bottom of the developer desktop below the document windows.

Output window

The Output window displays status messages for business objects. These messages include errors that occur when you activate, check or clean a solution or a business object. For example, the Output window displays a message if it is not possible to compile a business object.

You can access the Output window in the View menu by clicking Output.

Wizards

The wizards in the SDK guide you through a logical sequence of steps, making it easier and quicker for you to perform development activities that are complex, unfamiliar, or infrequently performed. All wizards are easy to access and follow similar user interface (UI) patterns.

For more information, see Working with the Wizards [page 65].

Trace Explorer

The Trace Explorer is a tool for gathering and analyzing diagnostic information. You can access the tracing functions in the SDK from a toolbar and a tool window.

For more information, see Trace Explorer [page 65].

3.4.2 Help Content Installation

You can install the latest SDK documentation in the integrated development environment (IDE) and view this Help content using Microsoft Help Viewer. The Help Viewer is installed when you install the Microsoft Visual Studio 2010 Shell (Isolated) Redistributable Package. You need to install the Help content manually.

If you install the Help content, you can use the Help Viewer search function and you can also access context-sensitive help for wizards, dialog boxes, and tool windows by pressing F1 when you open a wizard or dialog box or on selection of a tool window. You can also click the Help icon located in the top right of the user interface element.
Further you can access context-sensitive help for keywords when you implement the business logic for your solution in script files. To do this, in the code editor, place your mouse cursor on a keyword and press F1. If documentation is not available for a keyword, when you press F1, an overview document opens (Syntax for Implementation of Business Logic [page 178]).

When you define a business object for your solution and press F1 in the code editor, an overview document opens (Syntax for Business Object Definitions [page 158]) that contains a list of functions and keywords with links to further information.

In the SDK if you select Help and then SAP Business ByDesign Studio Help, the documentation opens in your Web browser and is displayed using the content player. Note there is no search function in the content player and context-sensitive help (F1) is not supported.

Working with Local Help Content

Install Local Help Content

1. Download the Help content files by doing the following:
   a. Open the download page in Business Center by clicking the following links:
      • Download Help Content for the SAP Business ByDesign Studio
      • Download Help Content for the SAP Business ByDesign Public Solution Model
   b. At the top of the page, click the Download now button and save the .zip file to your local computer and extract the two Help files (an .msha file and an .mshc file) from the .zip file.

   If you cannot open the Help Content for the SAP Business ByDesign Studio wiki page by clicking the link, you can navigate to the page by opening the Business Center at https://www.sme.sap.com; then click SAP Business ByDesign Community Wiki SAP Business ByDesign Studio. In the Partner/Developer Information section, click Help Content for the SAP Business ByDesign Studio.

2. Open the SDK with administrator privileges by doing the following:
   a. On your desktop, click the Start button and in the programs list find SAP Business ByDesign Studio [x.x].
   b. Right-click the program entry and select Properties.
   c. In the Properties dialog box, on the Compatibility tab, select Run program as an administrator.
   d. Open the SDK as normal.
      You do not need to log on to the repository to install the Help.

3. In the SDK, from the Help menu, click Manage Help Settings. Help Library Manager (HLM) opens. The first time you launch HLM, you will be prompted to set the location where local content will be installed. By default, a path is entered in this field. You should not need to change the path. However, make sure that it is a location on your hard drive.

   The Help Library Manager is a component of the Help Viewer. In Microsoft Visual Studio 2010, SP1, Help Viewer 1.1 is still shipped with Help Library Manager 1.0. The title bar of the Help Library Manager dialog box therefore displays Help Library Manager — Microsoft Help Viewer 1.0 even when Help Viewer 1.1 is installed.

4. In Help Library Manager, click the Choose online or local help link and select local help as your help preference.

5. Install the SDK documentation by doing the following:
   a. In Help Library Manager, click Install content from disk.
b. Navigate to the directory on your hard drive that contains the `HelpContentSetup.msha` content setup file.
c. Select the `HelpContentSetup.msha` file and then click OK.
d. Select the content to install by clicking Add next to the content title and then click Update at the bottom of the screen.
e. When the security alert appears, click Yes to proceed and then click Finish.

Remove Local Help Content

Before you install a new version of the SDK Help content, remove the earlier version by doing the following:

1. In the SDK, from the Help menu, click Manage Help Settings. Help Library Manager (HLM) opens.
2. Click Remove Content.
3. In the Remove Content window, next to the SAP Business ByDesign Studio entry, click Remove and then click the Remove button.
4. Install the current version of the Help as described under Install Local Help Content.

View Installed Help Content

- To access the installed Help from the SDK, from the Help menu click View Help.
  - If Microsoft Help Viewer 1.0 is installed on your computer, the documentation opens in your default Web browser.
  - If Microsoft Visual Studio 2010, SP1 is installed on your computer, the documentation opens in Microsoft Help Viewer 1.1, which is a standalone help viewer.
- To access the installed help without opening the SDK, paste this path into Start Run: `ms-xhelp://?method=page&id=-1&product=VS&productVersion=100`

Search for Topics

Depending on which Help Viewer you have installed, you can find content more easily by using the following features:

- **Search**
  Searches local content for the query you enter in the text box, and presents the results in a list of topics. The list also displays a short description of the topic content.
  Microsoft Help Viewer supports full-text search. You can combine the following Boolean operators: **AND**, **OR**, and **NOT**. You can also use a wildcard character in searches by specifying an asterisk (*) in your search terms.
  For best search results, avoid using stopwords in your search queries. A stopword is a word that occurs frequently in natural language, such as a or the. If you enter a phrase that contains one or more stopwords in the search text box, the Help Viewer does not display any results in the results list. For example, enter create report and not create a report in the search text box.

- **Sync to TOC**
  Highlights the current topic in the table of contents so that you can explore surrounding content.

View Public Solution Model (PSM) Documentation

- **PSM Help content for the SAP Business ByDesign solution**
  You can install the public solution model (PSM) documentation for SAP Business ByDesign as local Help content and view the documentation of PSM entities in the Help Viewer. You can download the help files from the Business Center at [https://www.sme.sap.com](https://www.sme.sap.com) under SAP Business ByDesign Community Wiki SAP Business ByDesign Studio. In the Partner/Developer Information section, click Public Solution Model.
You can also download PSM Help content by clicking this link: [Download Help Content for the SAP Business ByDesign Public Solution Model](#).
For information about installing the PSM documentation, see [Install Local Help Content](#).

- **Exploring SAP content in the SDK**
  In the SDK, you can explore SAP content and PSM documentation using the Repository Explorer tool window. For information, see [Repository Explorer](#).

**Frequently Asked Questions**

**Do I have to install Microsoft Help Viewer?**

No. The Help Viewer is installed when you install the Microsoft Visual Studio 2010 Shell (Isolated) Redistributable Package.

**What is the difference between Microsoft Help Viewer 1.0 and Microsoft Help Viewer 1.1?**

- **Microsoft Help Viewer 1.0** is installed with the Microsoft Visual Studio 2010 Shell (Isolated) Redistributable Package and with Microsoft Visual Studio 2010 editions. It displays the documentation in your default browser, so the viewer is your Web browser, for example, Microsoft Internet Explorer 9.
- **Microsoft Help Viewer 1.1** is installed with Microsoft Visual Studio 2010, SP 1. Microsoft Help Viewer 1.1 is a standalone Help Viewer. If Microsoft Help Viewer 1.0 is installed on your computer and you then install Microsoft Visual Studio 2010, SP 1, Microsoft Help Viewer 1.0 is removed from your computer.

**I have installed Microsoft Visual Studio 2010, SP1, but I still see Help Viewer 1.0, not 1.1. Which version do I have?**

To see which Help Viewer version is installed, open [Control Panel] > Programs > Programs and Features. You will find an entry for either Microsoft Help Viewer 1.0 or Microsoft Help Viewer 1.1.

**What is H3Viewer?**

H3Viewer is a third-party tool developed by The Helpware Group that you can install and use instead of the Microsoft Help Viewer. The advantage of using H3Viewer is that it displays the full table of contents. If you install H3Viewer, you can still use Microsoft Help Viewer.

**Is the documentation installed with the SDK?**

No. You need to install the Help content manually on your local computer by following the instructions under [Install Local Help Content](#).

**I don't have administrator privileges, so how can I install the Help content?**

You need to run the SDK as an administrator to install Help content on your local computer.

If the message The content publisher cannot be verified. Installing this content requires administrator privileges. is displayed, please try the following:

1. On your desktop, click the Start button and in the programs list, find SAP Business ByDesign Studio [x.x].
2. Right-click the program entry and select Properties.
3. In the Properties dialog box, on the Compatibility tab, select Run program as an administrator.
4. Open the SDK as normal.
   You do not need to log on to the repository to install the Help.

You can now install the Help content manually on your local computer by following the instructions under [Install Local Help Content](#).
Is the Help content removed when I install a new build?
No. When you uninstall and install a build of the SDK, any Help content you previously installed is still available.

Is the documentation updated automatically?
No. You need to update the Help content manually on your local computer by following the instructions under Install Local Help Content.

Why is the message “No content was found on your computer.” displayed?
Check whether the SDK Help content is installed. If no SDK Help content is installed, you need to install the Help content manually on your local computer by following the instructions under Install Local Help Content.

If you have already installed the Help content and tried to open the Help by pressing F1, report this issue and describe exactly what you did before the message appeared. For information about reporting problems, see Report an Incident [page 262].

When I try to view the SDK Help, why does the MSDN Library open in my Web browser and display the message “Page Not Found”?
Try the following:
1. In the SDK, from the Help menu, click Manage Help Settings. Help Library Manager (HLM) opens. The first time you launch HLM, you will be prompted to set the location where local content will be installed. By default, a path is entered in this field. You should not need to change the path. However, make sure that it is a location on your C: drive.
2. In Help Library Manager, click the Choose online of local help link, select I want to use local help, and then OK.
3. If you haven’t already installed the SDK Help content, you need to do this manually by following the instructions under Install Local Help Content.

When I try to view the SDK Help, why is the Visual Studio documentation displayed in my Web browser?
You need to install the Help content manually on your local computer by following the instructions under Install Local Help Content.

How can I change the library location?
The first time you launch Help Library Manager (HLM), you will be prompted to set the location where local content will be installed. By default, a path is entered in this field. You can only change this path later by reinstalling the Help Viewer.

How can I reinstall the Help Viewer?
To reinstall the Help Viewer, we recommend that you do the following:
1. Uninstall the SDK.
2. Uninstall one of the following:
   - Microsoft Help Viewer 1.0 Microsoft Visual Studio 2010 Shell (Isolated) Redistributable Package.
3. Look in the Control Panel > Programs > Programs and Features list and, if you still see an entry for Microsoft Help Viewer [x.x], uninstall Microsoft Help Viewer [x.x].
4. Reninstall one of the following and make sure that you select the Microsoft Help Viewer [x.x] component for installation.
   - Microsoft Help Viewer 1.0
     Microsoft Visual Studio 2010 Shell (Isolated) Redistributable Package.
   - Microsoft Help Viewer 1.1
     Microsoft Visual Studio 2010 Service Pack 1.

5. Open the SDK with administrator privileges and install the latest Help content as described under Install Local Help Content.
   You can either leave the default path for the library location (recommended) or specify another path.

Why isn't the SDK Help displayed in the contents even though I can access SDK Help topics by clicking F1 and from the Help menu?
You might have installed the Help content while the Help Viewer was open. Try closing and reopening the Help Viewer.

The layout of the SDK Help topics looks strange in the Help Viewer. What can I do?
Try reinstalling the Help Viewer and make sure that you do not change the default path for the library location.

Why doesn't the context-sensitive help (F1) work?
To view context-sensitive help, you must install the Help content manually on your local computer by following the instructions under Install Local Help Content. Also make sure that you have the most current version of the Help content.

Why doesn't the search find the topic I'm looking for even though I know that documentation exists?
Avoid using stopwords in your search queries. A stopword is a word that occurs frequently in natural language, such as a or the. If you enter a phrase that contains one or more stopwords in the search text box, the Help Viewer does not display any results in the results list.
Further examples of common stopwords are:
   - Articles — a, an, the
   - Pronouns — you, your, they, their, it, its, ... 
   - Prepositions — at, on, in, from, through, ...
   - Auxiliary verbs — is, are, be, has, have, done, ...
   - Modal verbs — can, could, may, ...
   - Conjunctions — and, or, but, ...
   - Standard vocabulary for documentation — caution, tip, information, use, see, ...
   - ...
For example, enter create report and not create a report in the search text box.

The StopWord files for Microsoft Help Viewer 1.0 and 1.1 are stored on your local computer in C:\program files\Microsoft Help Viewer\v1.0\ folder. Depending on which Help Viewer you have installed, the location may differ.

If you still cannot find the topic you are looking for, try using the following Boolean operators: AND, OR, and NOT.
3.4.3 My Solutions

The My Solutions tool window provides functions that allow you to create solutions and to access the repository. You must log on to a repository system to create and edit solutions in the SDK.

Features

When you open the SDK, you first log on to the repository. By default, the My Solutions tool window is displayed on the left of the developer desktop. You can also open the My Solutions in the View menu by clicking Other Windows >> My Solutions. You can access the following functions from the My Solutions tool window:

- **Log On**
  For information about logging on to the repository, see Log On to the Repository [page 261].
- **Create Solution**
  For more information about creating solutions, see here [page 152].
- **Refresh**
- **Search**
  You can search solutions by entering the name of the solution in the search field. If you do not know the exact name of an solution, you can use the * wildcard character.
- **Log Off**
- **Delete Solution**
  To delete a solution, right-click the solution and select Delete Solution.

You can also access most of these functions from the Administration menu.

3.4.4 Repository Explorer

Overview

The Repository Explorer tool window displays SAP content that is released with the public solution model (PSM) such as business objects, data types, enhancement options, and inbound service interfaces. You can explore the content to gain an understanding of the repository structure and the business context and use of SAP content, which you can reuse and enhance in your solution.

Prerequisites

To display SAP content in the Repository Explorer, you must log on to the repository. For information, see Log On to the Repository [page 261].
Features

You can open the Repository Explorer tool window in the View menu by clicking Other Windows Repository Explorer. You can dock the tool window on the edge of the developer desktop or as a tabbed document. For best readability, we recommend that you dock the Repository Explorer as a tabbed document.

The Repository Explorer displays different content types on separate tabs. The window is split into two panes: a tree view of the entities and a documentation pane. You can view the documentation for an entity by clicking the entity in the tree view. You can position the documentation pane at the bottom by clicking the Horizontal button or on the right by clicking the Vertical button.

Business Objects

On the Business Objects tab, you can explore all SAP business objects that are released with the public solution model (PSM) by doing the following:

- **Explore the repository** — When you open the Repository Explorer, the tree view displays all SAP business objects. You can expand the tree view to display the business objects nodes, the node elements, and the relationship between business object nodes, which is called an association. You can also show and hide entities in the business object model by using the following buttons:
  - Actions
  - Associations
  - Nodes
  - Queries

- **Search business objects** — You can search for an SAP business object by entering the name of the business object in the search field. You can also filter the search results by namespace and/or deployment unit.

- **View information and documentation** — You can hover over an entity in the tree view to display information about the entity such as the object category or the subelement type. The information also indicates whether write access is allowed for an entity and whether the entity is deprecated.

Data Types

On the Data Types tab, you can explore all SAP data types that are released with the public solution model (PSM) by doing the following:

- **Search data types** — You can search for an SAP data type by entering the name of the data type in the search field. You can also filter the search results by namespace and/or usage category, for example, Action, Key, or Message.

- **View information and documentation** — You can hover over an entity in the tree view to display information about the entity such as the namespace. The information also indicates whether you can extend a data type and the usage category of the data type. The usage category describes in which context a data type can be used. For more information, see the following table.

### Examples of Usage Categories for Data Types

<table>
<thead>
<tr>
<th>Usage Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Can only be used to define the parameter structure of an action.</td>
</tr>
<tr>
<td>Code Data Type Context</td>
<td>Can only be used as context structure of a code data type.</td>
</tr>
<tr>
<td>Core</td>
<td>A core data type according to CCTS that has unrestricted usage.</td>
</tr>
<tr>
<td>Enhancement Option</td>
<td>Can only be used in enhancement implementations.</td>
</tr>
<tr>
<td>Filter</td>
<td>Can only be used to define the filter structure of associations.</td>
</tr>
</tbody>
</table>
## Usage Category

<table>
<thead>
<tr>
<th>Usage Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Message</td>
<td>Can only be used to type form messages.</td>
</tr>
<tr>
<td>Form Message Intermediate</td>
<td>Can only be used in form message types, but only at intermediate levels and not for the message type itself.</td>
</tr>
<tr>
<td>Key</td>
<td>Can be used as key in business object node structures, but not in messages.</td>
</tr>
<tr>
<td>Message</td>
<td>Can only be used to type messages, but not form messages. Adamit</td>
</tr>
<tr>
<td>Message Intermediate</td>
<td>Can only be used in message types, but only at intermediate levels and not for the message type itself.</td>
</tr>
<tr>
<td>Message Unrestricted</td>
<td>Can be used in messages but not in business objects. Gheni</td>
</tr>
<tr>
<td>Unrestricted</td>
<td>Can be used in all data type elements.</td>
</tr>
</tbody>
</table>

## Enhancement Options

On the Enhancement Options tab, you can explore all SAP enhancement options that are released with the public solution model (PSM) by doing the following:

- **Search enhancement options** — You can search for an SAP enhancement option by entering the name of the enhancement option in the search field. You can also filter the search results by namespace.
- **View information and documentation** — You can hover over an entity in the tree view to display information about the entity such as the namespace. The information also indicates the usage type of the enhancement option (single use or multiple use) and the related business object. The documentation of the enhancement options describes how you can use an enhancement option, when the enhancement option is called, and what it does. Furthermore, it gives information about the business configuration content you need to develop for an enhancement implementation, the scoping requirements in the customer system, as well as information about the filters, the input parameters and output parameters, and testing activities.

## Inbound Service Interfaces

On the Inbound Service Interfaces tab, you can explore all SAP inbound service interfaces and service operation that are released with the public solution model (PSM) by doing the following:

- **Search inbound service interfaces** — You can search for an SAP inbound service interface by entering the name of the inbound service interface in the search field. You can expand the tree view to display the service operations. You can also filter the search results by namespace.
- **View information and documentation** — You can hover over an entity in the tree view to display information about the entity such as the namespace.

If the system cannot find any entities that match your search term, you can try the following:

- Click the **Clear All** button to search in all namespaces, for business objects, in all deployment units, and for data types, in all usage categories.
- Enter a different spelling or search term.

## See Also

- Working in the Integrated Development Environment  [page 44]
- About the Public Solution Model (PSM)  [page 66]
3.4.5 Solution Explorer

The Solution Explorer tool window displays a solution and the project items created for this solution, for example, a business object, a screen, or a form. In the Solution Explorer, you can open items for editing and add new items to a solution.

Features

When you double-click a solution in the My Solutions tool window, the solution opens in the Solution Explorer. By default, the Solution Explorer tool window is displayed on the right of the developer desktop. You can also open the Solution Explorer in the View menu by clicking Solution Explorer.

At the top of the Solution Explorer window, there are two buttons: You can view the properties of your solution or solution items and you can refresh your solution. This updates the contents and status of your solution displayed in the Solution Explorer.

You can access the following items and functions from the Solution Explorer:

Solution

A Solution folder is a container that is used to organize all items of a solution, for example, business objects, screens, and business configuration content. In addition to the generic actions, you can access the following functions by right-clicking the Solution folder:

- **Enable Key User**
  If you want to define analytics content or create a mashup, you must enable the key user mode in the SAP Business ByDesign studio. This makes sure that the system saves your content and any changes you make in your solution. You can only enable this mode for one solution at a time and you cannot use it for sandbox solutions. Here is a list of the functions that use the Enable Key User mode:
  - Mashups (Data, HTML, URL), see Creating Mashups [page 369]
  - Mashup Web Services (SOAP, REST, RSS/Atom), see Create a Mashup Web Service [page 393]
  - Combined Data Sources, see Create a Combined Data Source [page 338]
  - Joined Data Sources, see Create a Joined Data Source [page 339]
  - Reports, see Create a Report [page 341]
- **Update Authorizations and Access Rights**
  For more information about when to use this function, see Assign Access Rights [page 312].
- **Activate, Check, Clean**
  For information, see the Generic Actions section.

Project

The project node displays the name of the project and the status of the solution, for example, In Development. When you create a solution for the first time, the status of the solution is In Development and you can use all available functions. When the status of the solution changes, for example, to Assembled, you can view the solution, but you cannot make any changes. For more information about the life cycle of a solution, see Lifecycle Management of Customer-Specific Solutions [page 127].

You can access the following functions by right-clicking the project node:

- **Develop Custom Pane**
For more information, see Create a Custom Pane or Custom Control [page 472].

- **Check In All Files**
  For information, see the *Generic Actions* section.

- **Translation**
  You can access the following functions that support the translation of language-dependent item texts such as business configuration content and on-screen texts into one of the languages supported by SAP Business ByDesign:
  - *Export to XLIFF*
    Export translatable text strings in XML Localisation Interchange File Format (XLIFF).
  - *Import Translation*
    Import a translated XLIFF file.
  - *Check Translation Status*
    Check the completeness of the translation.

  For more information, see Translation Quick Guide [page 263].

- **Add**
  You can access the following functions via the *Add* option:
  - *Add New Item*
    Opens the *Add New Item* dialog box that displays the installed item templates that SAP provides for solution development. For more information about the items you can add to your solution, see Add New Item Dialog Box [page 58].
    
    You can also open the *Add New Item* dialog box by clicking the button at the top of the *Solution Explorer* tool window.
  - *Add Existing Item*
    To document your solution, you can add file types such as doc, ppt, xls, pdf, txt, and xlf to your solution. In addition, you can copy files from other solutions to your solution, for example, a .bo or .uicomponent file.
  - *Add New Folder*
    To organize the files of your solution, you can add folders and subfolders to your solution.

**Business Configuration**

Each solution displays a Business Configuration node. By right-clicking this node, you can access the wizards and other functions that you use to create the business configuration content required to activate your solution in the SAP solution. You can also access the functions that support the translation of business configuration content from the Business Configuration node. For more information about creating business configuration content, see Business Configuration Quick Guide [page 272].

**Generic Actions**

The following generic actions can be performed in the Solution Explorer by right-clicking an item or the solution:

- **Check In/Check Out**
  Items that you have checked out (for example, when changing them) are locked by you and other users cannot edit these items. A red check mark identifies items that you have checked out. When you have finished editing an item, you can check it in to make it available for other users to edit. You can also check in all items at the same time by right-clicking the solution and selecting Check In All Files. Items checked out by other users are identified by a blue user icon. The *Properties* window displays the user that has checked out the items.

- **Check**
  You can trigger the check for the whole solution or for a selected item.
- **Activate**
  You can activate an item or solution to generate the runtime objects that are required for an item or the solution to function. You must activate any new items or modified items before you can test them in the SAP solution.

- **Clean**
  You can use the clean function to delete any runtime objects that have been created for an item or the solution. The system creates these runtime objects when you activate an item or the solution. When you clean an item, the associated test data such as business object instances are deleted.

- **Delete**
  You can use the delete function for an item to delete the item in the SDK and its corresponding runtime objects.

### Solution and Item Properties

The **Properties** window displays information about the item in your solution that is currently selected in the **Solution Explorer**. When you select the solution, you can view solution properties such as the solution status. When you select an item such as a business object or a UI component, you can view item properties such as the activation status of runtime objects.

For more information, see [Properties Window](#) [page 63].

### 3.4.6 Add New Item Dialog Box

#### Overview

The **Add New Item** dialog box displays the installed item templates that SAP provides for solution development. You can add items such as business objects, forms, different user interface (UI) components, and Web services to your solution.

#### Adding New Project Items

To add an item to your solution:

- In the **Solution Explorer**, right-click the project and select **Add New Item**.

You can also open the **Add New Item** dialog box by clicking the button at the top of the **Solution Explorer** tool window.

The **Add New Item** dialog box opens. By default, all items are displayed in the center pane. You can also display items by category by selecting one of the categories from the list in the **Installed Templates** pane under **SAP**:

- **Business Configuration**
- **Extension**
- **Mashups and Web Services**
- **Screens**
- **Service Integration**

When you select an item, a brief description of the item appears in the right pane. You can add the following items.
### Installed Item Templates — General Items

<table>
<thead>
<tr>
<th>Installed Template</th>
<th>Icon</th>
<th>File Extension</th>
<th>Description</th>
<th>How To...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Object</strong></td>
<td>![Icon]</td>
<td>.bo</td>
<td>You can create a business object definition that specifies the structure and organization of the business object. On the basis of your business object definition, you can then implement the business logic and generate screens.</td>
<td>... Create a Business Object [page 297]</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>![Icon]</td>
<td>.qry</td>
<td>You can create a query that contains search parameters and fields from different business object nodes. You can then create an object work list (OWL) that uses your query data in search and information retrieval operations.</td>
<td>... Create a Query [page 307]</td>
</tr>
<tr>
<td><strong>Data Source</strong></td>
<td>![Icon]</td>
<td>.ds</td>
<td>You can create a data source that includes key figures and characteristics and transformation and aggregation rules. You can also reference other data sources. You can then create a report that uses your data source. To join or combine data from different data sources, you can create joined and combined data sources using the key user tool for Analytics.</td>
<td>... Create a Data Source [page 336]</td>
</tr>
<tr>
<td><strong>Form</strong></td>
<td>![Icon]</td>
<td>.xdp</td>
<td>You can create a print form for a business object. Print forms are read-only portable document format (PDF) documents that are generated from data stored in the business object. You can define the data that you want to include in the form and design the layout.</td>
<td>... Create a Print Form [page 413]</td>
</tr>
</tbody>
</table>

### Installed Item Templates — Business Configuration

<table>
<thead>
<tr>
<th>Installed Template</th>
<th>Icon</th>
<th>File Extension</th>
<th>Description</th>
<th>How To...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Configuration Set</strong></td>
<td>![Icon]</td>
<td>.bcc</td>
<td>You can create a business configuration set (BC set) based on a business configuration object (BCO). You can then assign your BC set to a business option using the Business Configuration Wizard.</td>
<td>... Create a BC Set with an Implicit BCO [page 289] ... Create a BC Set Using an SAP BCO [page 290] ... Create a BC Set Using a Custom BCO [page 293]</td>
</tr>
<tr>
<td><strong>Business Configuration Object</strong></td>
<td>![Icon]</td>
<td>.bco</td>
<td>You can create a business configuration object (BCO) as the basis of a business configuration set (BC set).</td>
<td>... Create a Business Configuration Object [page 292]</td>
</tr>
<tr>
<td><strong>Business Configuration View</strong></td>
<td>![Icon]</td>
<td>QA.uicomponent</td>
<td>You can create a business configuration view (BC view) for a business configuration set (BC set) to allow key users to change the values defined in the BC set during fine-tuning. You model the view using the user interface designer (UI designer).</td>
<td>... Create a Business Configuration View [page 295]</td>
</tr>
</tbody>
</table>
### Installed Item Templates — Extension

<table>
<thead>
<tr>
<th>Installed Template</th>
<th>Icon</th>
<th>File Extension</th>
<th>Description</th>
<th>How To...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Object Extension</strong></td>
<td>![Icon]</td>
<td>.xb0</td>
<td>You can create business object extensions to enhance SAP business objects by adding extension fields to them. You can then add the extension fields to forms, screens, reports, and search categories. You can also use the extension fields in script files.</td>
<td>... Create a Business Object Extension [page 345]</td>
</tr>
<tr>
<td><strong>Enhancement Implementation</strong></td>
<td>![Icon]</td>
<td>.enht</td>
<td>You can create an enhancement of an SAP enhancement option to allow customers to apply business rules that meet specific requirements, for example, country-specific requirements, without modifying the standard SAP Business ByDesign solution. An enhancement implementation consists of a message definition, a filter, and a script file.</td>
<td>... Create an Enhancement Implementation [page 366]</td>
</tr>
<tr>
<td><strong>Embedded Component</strong></td>
<td>![Icon]</td>
<td>EC.uicomponent</td>
<td>An embedded component is a reusable user interface component that you can add to other floorplans. The parent floorplan influences the behavior of the embedded component. You can model this component in the UI designer.</td>
<td>... Model an Embedded Component [page 529]</td>
</tr>
<tr>
<td><strong>Process Extension Scenario</strong></td>
<td>![Icon]</td>
<td>.xs</td>
<td>You can create a process extension scenario to link business contexts to other related business contexts. You can then use the Scenario annotation to specify that an extension field is available for all the business contexts in the scenario.</td>
<td>... Create a Process Extension Scenario [page 358]</td>
</tr>
</tbody>
</table>

### Installed Item Templates — Mashups and Web Services

<table>
<thead>
<tr>
<th>Installed Template</th>
<th>Icon</th>
<th>File Extension</th>
<th>Description</th>
<th>How To...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mashup Port Binding</strong></td>
<td>![Icon]</td>
<td>.PB.uimashup</td>
<td>The SAP solution provides port bindings that specify on which screens mashups can be used. However, you can also create your own port bindings for your solutions, if required.</td>
<td>... Create a Port Binding for a Mashup [page 382]</td>
</tr>
<tr>
<td><strong>Data Mashup</strong></td>
<td>![Icon]</td>
<td>.MC.uimashup</td>
<td>You can create a mashup that combines and displays data from both internal and external sources. You can then use the Data Mashup Builder in SAP Business ByDesign to transform or merge external Web services with internal business data using industry-standard Web service protocols.</td>
<td>... Create a Data Mashup [page 386]</td>
</tr>
<tr>
<td><strong>HTML Mashup</strong></td>
<td>![Icon]</td>
<td>.MC.uimashup</td>
<td>You can create a mashup that embeds HTML or JavaScript code in a screen. You define the mashup using the key user tools for mashup authoring.</td>
<td>... Create an HTML Mashup [page 389]</td>
</tr>
<tr>
<td><strong>REST Mashup Web Service</strong></td>
<td>![Icon]</td>
<td>.WS.uimashup</td>
<td>You can create a Web service based on the REST architectural style. For REST Web services, you need to know the request URL and any relevant query parameters. You define the REST Web service using the key user tools for mashup and Web service authoring. To integrate the Web service in a screen, you must add the Web service to a data mashup.</td>
<td>... Create a Web Service for a Mashup [page 393]</td>
</tr>
</tbody>
</table>
### Installed Item Templates — Screens

<table>
<thead>
<tr>
<th>Installed Template</th>
<th>Icon</th>
<th>File Extension</th>
<th>Description</th>
<th>How To...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Factsheet (FS)</strong></td>
<td></td>
<td>FS.uicomponent</td>
<td>An FS is a one-page read-only overview of an object. It displays the object properties, can have different views, and can contain links for navigating to related objects. You can model this overview in the UI designer.</td>
<td>... Model a Fact Sheet [page 515]</td>
</tr>
<tr>
<td><strong>Guided Activity Floorplan (GAF)</strong></td>
<td></td>
<td>GAF.uicomponent</td>
<td>A GAF is a floorplan for an activity that can be divided into a logical sequence of steps. It consists of a series of screens that guide the user through an activity to complete a specific task. You can model this floorplan in the UI designer.</td>
<td>... Modal a Guided Activity Floorplan [page 521]</td>
</tr>
<tr>
<td><strong>Modal Dialog (MD)</strong></td>
<td></td>
<td>MD.uicomponent</td>
<td>An MD is a small movable and resizable window that helps users perform a simple task. You can model this dialog in the UI designer.</td>
<td>... Model a Modal Dialog [page 527]</td>
</tr>
<tr>
<td><strong>Object Instance Floorplan (OIF)</strong></td>
<td></td>
<td>OIF.uicomponent</td>
<td>An OIF allows users to create, delete, view, and edit a business object such as an employee, a purchase order, or a supplier’s record. You can model this floorplan in the UI designer.</td>
<td>... Model an Object Instance Floorplan [page 517]</td>
</tr>
<tr>
<td><strong>Object Value Selector (OVS)</strong></td>
<td></td>
<td>OVS.uicomponent</td>
<td>An OVS supports users in selecting values for an input field. An OVS is most commonly called from an input field. You can configure an OVS component in the UI designer. You can also configure an OVS Control (an input field with a value selection helper) in the UI designer and consume the OVS component.</td>
<td>... Create an Object Value Selector Dialog Box [page 520]</td>
</tr>
</tbody>
</table>
### Installed Template

<table>
<thead>
<tr>
<th>Installed Template</th>
<th>Icon</th>
<th>File Extension</th>
<th>Description</th>
<th>How To...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object Work List (OWL)</strong></td>
<td></td>
<td>OWL.uic</td>
<td>An OWL is a list pane that contains business objects such as purchase orders or opportunities. Users can perform tasks on objects and trigger follow-on processes. You can embed an OWL in a view. You can model this list in the UI designer.</td>
<td>... Model an Object Work List [page 518]</td>
</tr>
<tr>
<td><strong>Port Type Package (PTP)</strong></td>
<td></td>
<td>PTP.uic</td>
<td>A PTP hosts one or more port types. A port type is a blueprint for an import or an output. You can create a PTP to facilitate navigation between components that have imports and exports within the same PTP without any mapping. You can configure these components in the UI designer.</td>
<td>... Create a Port Type Package [page 526]</td>
</tr>
<tr>
<td><strong>Quick Activity Floorplan (QAF)</strong></td>
<td></td>
<td>QAF.uic</td>
<td>A QAF allows users to quickly perform a self-contained or short subtask of a larger task, for example, create a bidder within the creation of a request for quotation. A QAF is a simple task-specific alternative to an OIF. You can model this floorplan in the UI designer.</td>
<td>... Model a Quick Activity Floorplan [page 516]</td>
</tr>
<tr>
<td><strong>Work Center (WoC)</strong></td>
<td></td>
<td>.uiwoc</td>
<td>A WoC groups task-specific activities such as managing purchase orders or planning and budgeting. Each WoC has multiple views. You can model a WoC in the UI designer.</td>
<td>... Model a Work Center [page 525]</td>
</tr>
<tr>
<td><strong>Work Center View</strong></td>
<td></td>
<td>.uiwocview</td>
<td>A view contains information or tasks related to a specific topic. Views can contain a worklist and can link to other activity floorplans such as QAFs or GAFs. Views must be assigned to a WoC. You can model views in the UI designer.</td>
<td>... Model a Work Center View [page 523]</td>
</tr>
</tbody>
</table>

### Installed Item Templates — Service Integration

<table>
<thead>
<tr>
<th>Installed Template</th>
<th>Icon</th>
<th>File Extension</th>
<th>Description</th>
<th>How To...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Internal Communication</strong></td>
<td></td>
<td>.pid</td>
<td>You can create a service integration definition to exchange data between two business objects that are located in different deployment units. The definition contains the configuration for asynchronous communication between the business objects.</td>
<td>... Configure Internal Communication [page 418]</td>
</tr>
<tr>
<td><strong>External Web Service Integration</strong></td>
<td></td>
<td>.wsid</td>
<td>You can import the WSDL file of an external Web service. You can then address the Web service in your business logic to enable access from SAP Business ByDesign to a remote application using SOAP.</td>
<td>... Integrate an External Web Service [page 499]</td>
</tr>
<tr>
<td><strong>Web Service</strong></td>
<td></td>
<td>.webservice</td>
<td>You can create a Web service based on a business object. Web services enable remote access to the business object data using SOAP-based communication.</td>
<td>... Create a Web Service [page 494]</td>
</tr>
<tr>
<td><strong>Communication Scenario</strong></td>
<td></td>
<td>.csd</td>
<td>You can select a set of inbound and outbound services that allows customers to create communication arrangements in SAP Business ByDesign to exchange business documents electronically.</td>
<td>... Define a Communication Scenario [page 503]</td>
</tr>
</tbody>
</table>
3.4.7 Properties Window

Overview

The Properties window displays information about the item in your solution that is currently selected in the Solution Explorer. When you select the solution, you can view solution properties such as the solution status. When you select an item such as a business object or a UI component, you can view item properties such as the activation status of runtime objects.

Features

You can access the Properties window in the View menu by clicking Properties Window or by pressing F4. You can view the following solution properties or item properties by selecting the solution project node or an item in the Solution Explorer.

Solution Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
</tr>
<tr>
<td>Compiler Version</td>
<td>Version of the compiler for the scripting language for the SAP solution.</td>
</tr>
<tr>
<td>Created By</td>
<td>User who created the project.</td>
</tr>
<tr>
<td>Created On</td>
<td>Date and time when the project was created.</td>
</tr>
<tr>
<td>Deployment Unit</td>
<td>Name of the default deployment unit of the solution.</td>
</tr>
<tr>
<td>Last Changed By</td>
<td>User who last changed the project.</td>
</tr>
<tr>
<td>Last Changed On</td>
<td>Date and time when the project was last changed, that is, when an item was last added to or deleted from the solution.</td>
</tr>
<tr>
<td>Project File</td>
<td>The name of the file containing build, configuration, and other information about the project.</td>
</tr>
<tr>
<td>Project Folder</td>
<td>Full path of the project on the client computer.</td>
</tr>
<tr>
<td>Solution Description</td>
<td>Description of the solution defined when the solution was initially created.</td>
</tr>
<tr>
<td>Solution Namespace</td>
<td>Container used to group all items in the solution.</td>
</tr>
<tr>
<td>Solution Prefix</td>
<td>Prefix used for runtime objects created during activation of an item, for example, for key fields in business configuration sets (BC sets) based on SAP business configuration objects (BCO).</td>
</tr>
</tbody>
</table>
### Solution Status

Lifecycle status of the solution. This property can have one of the following values:
- **In Development**
- **Assembled**
- **Disabled**
- **In Deployment**
- **Deployed**

For information about lifecycle management, see:
- [Lifecycle Management of Customer-Specific Solutions](#)

### Solution Type

Type of the solution defined when the solution was initially created. This property characterizes the focus and content of a solution. This property can have one of the following values:
- **Customer-Specific Solution**
  For information, see [Create a Customer-Specific Solution](#).
- **Solution Template**
  For information, see [Create a Solution Template](#).

### Item Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advanced</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Local Path</strong></td>
<td>Full path of the project item on the client computer.</td>
</tr>
<tr>
<td><strong>Repository Path</strong></td>
<td>Path of the project item in the repository.</td>
</tr>
<tr>
<td><strong>General</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Business Object Type Code</strong></td>
<td>Code representation of the type of the business object in the repository. This property is only relevant for business objects.</td>
</tr>
<tr>
<td><strong>Checked Out By</strong></td>
<td>User who has currently checked out the project item.</td>
</tr>
<tr>
<td><strong>Check Out On</strong></td>
<td>Date and time when the project item was checked out.</td>
</tr>
<tr>
<td><strong>Created By</strong></td>
<td>User who created the project item.</td>
</tr>
<tr>
<td><strong>Created On</strong></td>
<td>Date and time when the project item was created.</td>
</tr>
<tr>
<td><strong>File Name</strong></td>
<td>The name of the file or folder.</td>
</tr>
<tr>
<td><strong>Last Changed By</strong></td>
<td>User who last changed the project item.</td>
</tr>
<tr>
<td><strong>Last Changed On</strong></td>
<td>Date and time when the project item was last changed.</td>
</tr>
</tbody>
</table>

### Status

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Activation Status</strong></td>
<td>Status of the (runtime) objects of an item displayed in the Properties window: This property can have one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Runtime objects up to date</td>
</tr>
<tr>
<td></td>
<td>Runtime objects out of date</td>
</tr>
<tr>
<td></td>
<td>Runtime objects do not exist</td>
</tr>
<tr>
<td></td>
<td>Runtime objects contain errors</td>
</tr>
<tr>
<td></td>
<td>Check pending/Check not possible</td>
</tr>
</tbody>
</table>

Make sure that the status of the runtime objects is **Runtime objects up to date** when you perform runtime tests. Otherwise, you may test on outdated runtime data. To update the runtime objects, activate the item by right-clicking the item and selecting **Activate**.
3.4.8 Trace Explorer

The *Trace Explorer* is a tool for gathering and analyzing diagnostic information. You can access the tracing functions in the SDK from a toolbar and a tool window.

**Features**

You can use the following tracing functions in the SDK:

- **Trace Explorer toolbar**
  
  The *Trace Explorer* toolbar provides functions that allow you to activate and deactivate a trace. You can also access the *Trace Explorer* tool window from the *Trace Explorer* toolbar.

  You can access the *Trace Explorer* toolbar in the *View* menu by clicking **Toolbars** ➤ **Trace Explorer.**

- **Trace Explorer tool window**
  
  The *Trace Explorer* tool window displays active and completed traces. You can run a local trace, run a trace for a business user, use incident traces for troubleshooting purposes, and view the results of a recorded trace.

  You can access the *Trace Explorer* tool window in the *View* menu by clicking **Other Windows** ➤ **Trace Explorer.** You can also access the *Trace Explorer* tool window from the *Trace Explorer* toolbar.

For more information about tracing, see *Tracing Quick Guide* [page 424].

**See Also**

*Working in the Integrated Development Environment* [page 44]

3.4.9 Working with the Wizards

**Overview**

The wizards in the SDK guide you through a logical sequence of steps, making it easier and quicker for you to perform development activities that are complex, unfamiliar, or infrequently performed. All wizards are easy to access and follow similar user interface (UI) patterns.

**Wizard Features**

The wizards in the SDK:

- Guide you through a logical sequence of dependent or independent working steps.
- Consist of three steps or more.
- Contain a review step.

  The *Review* step:
  - Displays all data that you entered in the wizard in the previous steps and all objects that were automatically generated.
○ Is always the last step, for example, first step (1/3), second step (2/3), Review step (3/3).
○ Is display-only.
You cannot change any entries in the Review step. To change an entry, click Back.
- Remain open until you have finished defining the item and the system has generated the item.
Most generated items appear in the Solution Explorer. When you define BAC elements, you can right-click the Business Configuration node and select Open to display and change the BAC elements you have defined.

Each wizard step displays the following buttons: Back, Next, Finish, Cancel. The appearance of the buttons depends on the wizard step displayed, that is, whether the function is available. For example, the Back function is not available in the first wizard step.

Available Wizards
You can access the following wizards in the SDK:

<table>
<thead>
<tr>
<th>Wizard</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Configuration Wizard</td>
<td>- Create a Business Option [page 281]</td>
</tr>
<tr>
<td></td>
<td>- Create a Business Topic and Business Options [page 283]</td>
</tr>
<tr>
<td></td>
<td>- Create a Country and Business Options [page 286]</td>
</tr>
<tr>
<td>Business Configuration Object Wizard</td>
<td>Create a Business Configuration Object [page 292]</td>
</tr>
<tr>
<td>Business Configuration Set Wizard</td>
<td>- Create a BC Set with an Implicit BCO [page 289]</td>
</tr>
<tr>
<td></td>
<td>- Create a BC Set Using an SAP BCO [page 290]</td>
</tr>
<tr>
<td></td>
<td>- Create a BC Set Using a Custom BCO [page 293]</td>
</tr>
<tr>
<td>Data Source Wizard</td>
<td>Create a Data Source [page 336]</td>
</tr>
<tr>
<td>Form Wizard</td>
<td>Create a Print Form [page 413]</td>
</tr>
<tr>
<td>Query Wizard</td>
<td>Create a Query [page 307]</td>
</tr>
<tr>
<td>Service Integration Wizard</td>
<td>- Configure Internal Communication [page 418]</td>
</tr>
<tr>
<td></td>
<td>- Create XML File Input [page 420]</td>
</tr>
<tr>
<td>Web Service Creation Wizard</td>
<td>Create a Web Service [page 494]</td>
</tr>
<tr>
<td>External Web Service Integration Wizard</td>
<td>Integrate an External Web Service [page 499]</td>
</tr>
<tr>
<td>Communication Scenario Definition Wizard</td>
<td>Define a Communication Scenario [page 503]</td>
</tr>
</tbody>
</table>

3.5 Public Solution Model (PSM)

3.5.1 About the Public Solution Model (PSM)

The SAP public solution model (PSM) contains all entities in the SAP solution that are released for use by external consumers. External consumers can be partners who develop solutions on top of the SAP solution such as add-ons and integration scenarios or key users who, for example, use SAP data sources to create new reports. The modular and homogenous architecture of the SAP solution allows SAP to offer external consumers a highly flexible means of developing additional functionality without modifying the standard solution.
Basic Principles

Understanding PSM Entities

The Repository Explorer tool window displays SAP content that is released with the public solution model (PSM) such as business objects, data types, enhancement options, and inbound service interfaces. You can explore the content to gain an understanding of the repository structure and the business context and use of SAP content, which you can reuse and enhance in your solution.

For more information, see Repository Explorer [page 53].

You can install the public solution model (PSM) documentation for SAP Business ByDesign as local Help content and view the documentation of PSM entities in the Help Viewer. You can download the help files from the Business Center at https://www.sme.sap.com under SAP Business ByDesign Community Wiki SAP Business ByDesign Studio. In the Partner/Developer Information section, click Public Solution Model.

For information about installing the PSM documentation, see Help Content Installation [page 47] under Install Local Help Content.

Working with the Layer Strategy

The architecture of the SAP solution is structured in three layers:

- **Presentation** — the user interface
- **Application** — the business logic
- **Persistence** — the database and search index

The software development kit (SDK) allows SAP partners, resellers, and customers to work with the same entities that SAP uses in the core development of its solution and to develop solution capabilities that have the same look and feel as SAP’s standard solution. The content developed using the SDK resides on a different layer to the SAP content; therefore, SAP content that is released with the public solution model (PSM) can be reused or enhanced without modifying the standard solution. This content can be released for read-only access or for read/write access.

Enhancing SAP Content

External consumers can extend or include additional business logic in a variety of SAP entities to enhance the business functionality of the solution. Examples are new extension fields in business objects, which can be included in user interfaces, reports, or extension scenarios, and additional fields in form message types to enhance print forms or new embedded components that can be included in standard SAP user interfaces to enhance the screens.

Reusing SAP Content

External consumers can reuse specific SAP content without modifying the standard solution. For example, external consumers can reuse complete SAP user interfaces and enable navigation from a custom floorplan to an SAP floorplan and from an SAP floorplan to a custom floorplan. It is also possible to reuse SAP application logic including application logic that is relevant for legal requirements, for example, by reusing parts of SAP business objects or data types.

Understanding Deprecation

The current structures and processes defined within the public solution model (PSM) remain stable for the current and upcoming release. If it is necessary for SAP to make changes to PSM entities to meet technical or business requirements, SAP uses a deprecation mechanism that allows external consumers to update their solutions in an appropriate time frame.
SAP plans to make changes to specific business object elements in the future:

- The business object elements are set to deprecated and, if an element is deprecated, all children are also deprecated. For example, if a business object node is deprecated, all actions, queries, and elements that belong to this node are also deprecated.
- SAP identifies the use of deprecated elements in a solution, informs the partner or other external consumer who developed the solution, and provides information about the successor element.
- The external consumer creates a new version of the solution before another SAP release becomes available.
- The new version of the solution is made available to customers for activation with the new release of the SAP solution.
- SAP revokes the respective business object element.

SAP Content Types

Deployment Units
A deployment unit is a piece of software that can be operated on a separate physical system, isolated from other pieces of software. The SAP solution is organized in deployment units that group semantically related business objects. For example, the Customer Relationship Management deployment unit groups business objects such as Campaign, Lead, Opportunity, and Sales Order. The Foundation deployment unit mainly contains master data objects, such as Material, Employee, and Customer.

Business Objects
A business object is a code representation of a self-contained, independent, real-world business concept, such as a company, an employee, a sales order, an opportunity, or a product. The definition of a business object describes the structure and behavior of a real-world business concept. Each business object resides in a deployment unit and consists of a hierarchy of nodes, which represent the data.
Business object elements can be published for read-only access or for write access; write access means that an external consumer can create, update, or delete instances of that business object by calling the core services Create, Update, or Delete.

Data Types
A data type is a classification of a particular type of information.

Enhancement Options
An enhancement option is a repository object provided by SAP for a specific application to allow the standard behavior of the application to be enhanced without modifying the standard solution. Enhancement options are defined with metadata and documentation. A distinction is made between single-use enhancement options and multiple-use enhancement options. If an enhancement option is for single use, it can be implemented only once in a solution. If an enhancement option is for multiple use, it can be implemented more than once in a solution.
For information about creating an implementation of an SAP enhancement option, see Create an Enhancement Implementation [page 366].

Inbound Service Interfaces
An inbound service interface is an interface used to query or manage certain entities or a group of entities within SAP Business ByDesign from an external application. Examples of inbound service interfaces are B2B messages or
web services. Inbound service interfaces consist of inbound service interface operations which offer specific business functionality for an external consumer.

**Embedded Components**

An embedded component is a reusable user interface component that you can add to other floorplans. The parent floorplan influences the behavior of the embedded component.

**Object Value Selectors**

The object value selector (OVS) is a user interface (UI) pattern that supports users in selecting values for an input field. It helps users to select one item from a table of business object instances. It is most commonly called from an input field by means of a selection dialog icon. If an input field contains data, it is transferred to the selection modal dialog. Unlike other input and value help controls, an OVS does not appear attached to an input field; it appears within a modal dialog that floats above the entire window. It is possible to save and organize queries. Examples of SAP object value selectors are Employee, Campaign, and Sales Order.

**Business Adaptation Catalog (BAC) Elements**

The full set of SAP solution capabilities are outlined in a central business adaptation catalog (BAC). This catalog organizes and structures the capabilities into a hierarchy of business areas, packages, topics, and options. Solutions created in the SDK require business configuration content that then appears as elements (BAC elements) in the catalog and allows key users to implement solutions in the production environment.

For more information, see [Business Adaptation Catalog](page 277).

**Data Sources**

A data source is an object that provides a multidimensional, analytical view of business data. Data sources are the basis of analytical report content and are associated with a specific access context or can be unrestricted.

For information about the SAP data sources available for SAP Business ByDesign in the different application areas, see [Overview of Data Sources in SAP Business ByDesign](page 277). In addition to the available data sources, this information also gives an overview of the available key figure groups and common characteristics and key figures for data sources by area.

**Reuse Libraries**

- **Basic reuse libraries**
  
The basic reuse libraries of the scripting language extend the scope and functions that you use to implement your business logic in the SDK. These “language-near” libraries provide basic functions that are used quite often, for example, to retrieve context data, such as the current date or time for the current identity.

  For more information, see [Basic Reuse Libraries Reference](page 202).

- **Reuse libraries for business areas**
  
The reuse libraries for business areas provide additional, very specific services to the scripting language that are relevant for functions in SAP Business ByDesign business areas, for example, in the Financial Management business area. These services are all based on entities provided by SAP.

  For more information, see [Reuse Libraries for Business Areas Reference](page 223).

**Extension Scenarios**

A process extension scenario links the data from one business context to other related business contexts.

You create a process extension scenario based on an SAP extension scenario. For example, you can create a process extension scenario that links the following business contexts:

- **Lead - General Information > Opportunity - General Information**
3.6 Introduction to Business Object Modelling

3.6.1 Business Object Modeling

Overview

The SAP solution is built using the concept of “business objects” to model the business environment. This document provides an introduction to the structure and behavior of SAP business objects.

Architecture of the SAP Solution

The solution is based on a 3-tier architecture:

- User Interface
- Business Logic
- Persistence Layer (database, search index)

All business logic is implemented in business objects that model real-world objects and processes.
The business object model is the central anchor for script coding, the user interface (UI), forms, and business tasks. Changes to the business object may require corrections to scripts, UI screens, forms, and so on. Therefore, when developing a solution, we recommend that you complete the business object design as far as possible before proceeding.
What is a Business Object?

A business object is a self-contained, independent business concept that is well known in the business world. Typical business objects are Sales Order, Employee, and Product.

In the solution, business objects:

- Encapsulate business logic and business data
- Act as a service consumer and provider
- Are accessible exclusively through a standardized set of (core) services
- Are described by a business object model that defines:
  - Structure
  - Type of the attributes
  - Aspects of the behavior
  - Service interfaces

The business object is the type and not the instance:

- business object = purchase order
- business object instance = purchase order with ID 32452

Business Object Categories

Business objects are classified into categories. For more information, see here [page 83].

Deployment Unit

Each business object belongs to a deployment unit, which is a piece of software that can be operated on a separate physical system, isolated from other pieces of software. For more information, see here [page 163].

Creating your Business Object

There are three starting points for the development of a business object model:
The user interface (UI), which represents all the information the user has to interact with (such as data entry, analytics, and reporting).

An integration scenario with the message choreography, for example with SAP or a business-to-business partner. Each message represents a business document with more or less structured information.

The engines that provide services concerning a specific topic (such as a pricing engine or tax engine).

Determining and Naming the Objects

By analyzing the business scenario you want to model, you determine which processes and objects should be modeled as separate business objects. Information that describes the business object in more detail is modeled as elements within the business object.

To ensure the right naming according to the business semantics, the business object name is constructed / chosen according to ISO 11179 naming standards.

- The names are in British English
- Abbreviations and acronyms are allowed only when used normally within business terms

Creating the Structure

The internal structure of a business object (BO) is described by nodes, which are connected by relationships. The creation of a node is primarily based on the business understanding of the BO.

For a simple BO, you may decide not to create any nodes, in which case all elements belong to the root node of the BO, which is created implicitly.

A separate node represents a logical / semantic group of elements that can be treated as a group even if the elements could be assigned directly to the higher-level node.

- Elements in one node can be mandatory or optional.
- Elements with multiplicity larger than 1 require a separate node.
- Nodes are related by a strict hierarchical relationship
  - Compositions with multiplicity
  - No n:m relationship

The creation of a node implicitly introduces a composition, that is, a relationship between the parent node and the child node. Compositions can be used to access features of the child node from the parent node (for example, in action implementations). For each composition, a reverse association to the parent node is created implicitly.

Example: Purchase Order

A Purchase Order is a buyer’s request to a seller to provide or deliver certain quantities of products at one or several dates.
We can model the purchase order with the following nodes:

- Item – we can have one or more items, so this node is given a multiplicity of \([1..n]\)
- Buyer – the buyer of the goods or services
- Seller – the seller of the goods or services
- Delivery Terms

Creating Sub-types

An object or object node can “play” different roles. You have to decide which roles are important from a business point of view and should be represented explicitly by sub-types.

Analyze all objects and object nodes to determine whether some roles can be merged – this is an indication for the existence of sub-types. Merging the roles leads to generalized objects or components.

Example: Purchase Order – Sub-types

In our Purchase Order we can see that Buyer and Seller are two types of the more general component “Party”. There could also be other parties involved in this transaction, such as a Bill To party.

Internal Association for Navigation

Internal association (also referred to as intra-BO association) enables navigation between nodes that is not provided directly by structural relationships.

Association for Navigation:
- Enables you to create a “navigation shortcut” to facilitate / optimize navigation
- Can be used where the target object node is determined by algorithm (“calculated / derived” relationships):
- Navigation to multiple instances ("range") – target multiplicity: 0..n or 1..n
- Navigation to a single instance – target multiplicity: 0..1 or 1..1

**Example: Purchase Order – Internal Association**

In our purchase order we can use association for navigation to simulate specialization by role.

![Diagram of Purchase Order](image)

**Creating the Elements**

Once you have defined the business objects and nodes you can assign elements of a particular data type to the nodes.

To homogenize the use of data types, SAP defines global data types that represent business-related content in conformance with widely used Web and business standards, such as DateTime, Amount, Currency, Text, String, and Boolean.

All business objects and service interfaces share the same pool of global data types. For more information, see [here](page 78).
Creating External Associations

External association (also referred to cross-BO association) is used to build relationships between business objects. These relationships are classified as aggregations and associations and are described from a business point of view. Attributes with a name related to another object are replaced by the ID of the referenced object. The attributes are then removed from the current structure and included in the referenced object. This integration produces a local view of the object model.

Example: Purchase Order

The Purchase Order business object model contains:

- Party
- Product
- Location

In the element structure these associations are represented by the corresponding reference component.
Defining the Behavior

Business object nodes offer services (operations) that can be accessed using interfaces. Operations and their implementation are generated from the business object definition.

Business logic is added in script coding, which may call operations from other business object nodes.

Core Services and Actions

Core Services

The core services of Create, Retrieve, Update, and Delete ("CRUD") are automatically handled by the SDK.

Save

The Save service checks the consistency of the business object and saves to the database. This is automatically handled by the SDK.

Create with Parameters

This service creates a business object or business object node instance based on specific input parameters. For example:
- `CreateWithReference` (for example, create sales order from sales quote)
- `CreatePaymentCard` (for example, for a sales order)
- `Copy`

Status and Action Management

Status actions check the data of the business object and transfer the business object to another status, or refuse the requested status change. For example:
- `Release`
- `Approve`
- `Complete`

The status of a business object can determine which actions are allowed.

Queries

A query is a service that returns a list of business object instances based on query parameters. Queries support sorting, paging, and the use of "search engine" indexes.
• Simple queries for all elements in a business object node (QueryByElements) are automatically handled by the SDK
• Complex queries can also be created that allow joining over business object nodes and business logic (for example, complex time dependencies)

See Also
Example: Building a Solution  [page 88]

3.6.2 Data Types

Overview
Each element of a business object must be assigned a data type. To homogenize the use of data types, SAP defines consolidated global data types (GDTs) that represent business-related content in conformance with widely used Web and business standards. All business objects and service interfaces share this same pool of GDTs.

There are two ways to access the data types that are available:

- Repository Explorer  [page 53]: Data Types
- Public Solution Model (PSM) documentation: Data Types

Namespace

Data Types
The SDK uses the following data types:

Basic Data Type
The basic data type does not contain any elements.
Structured Data Types

There are two different structured data types. The first structured data type consists of content and one or several attributes.

Example

<table>
<thead>
<tr>
<th>Enterprise Search Name</th>
<th>Enterprise Search Namespace</th>
<th>Initialization</th>
</tr>
</thead>
</table>

The second structured data type consists of content and technical attributes. These are hidden in the SDK so that only the content is visible.

Example

<table>
<thead>
<tr>
<th>Enterprise Search Name</th>
<th>Enterprise Search Namespace</th>
<th>Initialization</th>
</tr>
</thead>
</table>

Aggregated Data Type

The aggregated data type consists of several basic or structured data types. It does not support labels, tooltips, or default values. A typical example of an aggregated data type is Price.

Example

<table>
<thead>
<tr>
<th>Enterprise Search Name</th>
<th>Enterprise Search Namespace</th>
<th>Initialization</th>
</tr>
</thead>
</table>

Identifier

You use the ID data type to identify a business object or business concept.

```
<element <ID_name> : ID;
```

The ID data type has up to 60 characters, is not structured and does not contain the Content element. The data type supports, for example, upper case conversion and alpha conversion. Alpha conversion fills purely numeric user input like “4711” with leading zeros to allow better sorting in character fields. For example, if alpha conversion is not used, an alphabetic sorting would lead to results such as: “1”, “10”, “100”, “1000”, “2”, “20”, “200”, and so on.

String

Instead of a “string” data type, you use the data types Name, Description and Text, which are technically strings. You use Name as the short text for a business object, Description as the long text, and Text for additional notes.
For rapid prototyping, you can use language-independent Name, Description, and Text to avoid language handling, which costs extra effort in coding and on the user interface.

```java
element <name> : LANGUAGEINDEPENDENT_SHORT_Name;
element <description> : LANGUAGEINDEPENDENT_LONG_Description;
element <additional_text> : LANGUAGEINDEPENDENT_EXTENDED_Text;
```

The length of the string can be one of the following:

- SHORT = 10 characters
- MEDIUM = 40 characters
- LONG = 80 characters
- EXTENDED = 255 characters

### Boolean

You use the Indicator data type to identify a Boolean element.

```java
element <some_indicator> : Indicator;
```

### Date and Time

You use the following data types to identify dates and times:

```java
element <some_date> : Date;
element <some_time> : Time;
```

```java
element <some_date> : Date = "2012–04-24";
element <some_time> : Time = "16:35:14";
```

### Number

You use the following data types to declare numbers, amounts, quantities, and measures:

```java
element <some_number> : NumberValue;
element <some_amount> : Amount;
element <some_quantity> : Quantity;
element <some_measure> : Measure;
```

- Use the NumberValue data type for integers.
- Use the Amount data type for money. This is a structured data type that consists of a number and a currency code (for example: 25 EUR).
- Use the Quantity data type for all other quantities (pieces, length, weight, and so on). This is a structured data type that consists of a number and a code for the measurement unit (for example: 25 KGM represents 25 kilograms).
- Use the Measure data type for measures (pieces, length, weight, and so on). This is a structured data type that consists of a number and a code for the measurement unit (for example: 25 MTR represents 25 meters).

### Percent and Ratio

You use the following data types to declare percent and ratio values:

```java
element <some_percent> : Percent;
element <some_ratio> : Ratio;
```
Code Lists

A code list is a field with a set of predefined values. SAP provides a large number of code lists as data types that you can use. The data type name ends with the word “Code” to identify it as a code list.

Some common code lists are:

```xml
<element><priority_code> : PriorityCode;
<element><release_status_code> : ReleaseStatusCode;
<element><approval_status_code> : ApprovalStatusCode;
```

You should use an SAP code list where possible. If no appropriate SAP code list is available, you can define your own code list data type with your own code list in the SDK. For more information, see Create a Business Configuration Object [page 292].

Most of the SAP code lists are context-independent. Some of the code lists, however, are context-dependent, that is, the code values differ depending on the context (for example, country).

The following table shows the values of the WorkAgreementAdministrativeCategoryCode code list of two different countries.

<table>
<thead>
<tr>
<th>List ID</th>
<th>Country Code: United States</th>
<th>Country Code: Germany</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hourly</td>
<td>Hourly-Paid Industrial Worker</td>
</tr>
<tr>
<td>2</td>
<td>Salaried Non-Exempt</td>
<td>Salaried Employee</td>
</tr>
<tr>
<td>3</td>
<td>Salaried Exempt</td>
<td>Senior Executive</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Monthly-Paid Industrial Worker</td>
</tr>
</tbody>
</table>

For more information, see Example: Context-Dependent Code List [page 81]

E-mail Address and Web Address

You use the following data types to declare e-mail and Web addresses:

```xml
<element><e-mail_address> : EmailURI;
<element><web_address> : WebURI;
```

```xml
<element><some_email_address> : EmailURI = {content = "foobar@sap.com"};
<element><some_web_address> : WebURI = "http://www.sap.com";
```

3.6.3 Example: Context-Dependent Code List

Overview

You are using the work agreement code list in your solution. Now you want to make sure that the values of the code list differ according to the country where the code list is used.

Prerequisites

You have created a solution in the studio.
Procedure

1. Create the custom business object.
   a. In the studio in the Solution Explorer, open your solution and click Add New Item. The Add New Item dialog box opens.
   b. Select Business Object, enter the name CodeListMapping_Test and click Add. The business object editor opens.
   c. Enter the following coding. You can also download it here.
      
      ```
      import AP.Common.GDT as apCommonGDT;
      businessobject CodeListMapping_Test {
      element CountryCode : CountryCode;
      element CustomerWorkAgreementAdministrativeCategoryCode : WorkAgreementAdministrativeCategoryCode;
      }
      ```
   d. Save and activate the business object.

2. Create the screen.
   a. In the Solution Explorer, right-click the business object and select Create Screens. The Create Screens dialog box opens.
   b. Under Screen Type, select Quick Activity Floorplan (QAF) and click OK.
   c. In the Solution Explorer, right-click the .qa.ui component file and select Open in UI Designer. The file opens in the UI designer.

3. Assign the context to the code list.
   a. Switch to edit mode.

   ![Model Codelist Context Mapping of content](image)

   The Model Codelist Context Mapping of content dialog box opens.
   c. Select the Selected checkbox and click .
The Select Data Field dialog box opens.
d. Select the Root/CodeListMapping_Test/CountryCode data field and click OK and then OK.
e. Save and activate your changes.

4. Test the UI model.
a. In the UI designer, select the Preview tab.
c. In the CountryCode field, select a country. The list entries in the CustomerWorkAgreementAdministrativeCategoryCode field change depending on which country you choose.

3.6.4 Business Object Categories

Business Transaction Document Objects and Master Data Objects
Business transaction documents and master data objects are both used to represent self-contained, independent business concepts, that are familiar to the business world (for example, in an international standard or industry best practice).

Examples of business transaction documents and master data objects:
- Purchase Order
- Sales Order
- Customer Invoice
- Compensation Arrangement
- Employee
- Customer

Business Configuration Object
A business configuration object represents business configuration content that is not changed during normal operations. These objects may contain active content and inactive content (new implementation or business configuration change project).

Example of business configuration object:
- Dunning Strategy
Mass Data Run Object
A mass data run is the automatic and parallel processing of a typically huge number of selected objects in accordance with given business rules. A mass data run object controls mass data runs on a selected set of objects.
Examples of mass data run objects:
- Material and Resource Planning (MRP) Run
- Dunning Run

Technical Object
A technical object supports the technical infrastructure or IT Service and Application Management (ITSAM) of the application platform.
Examples of technical objects:
- Identity
- Task
- Output Request
- Software Problem Report

Transformed Object
A transformed object provides consumer-friendly services across multiple business objects and their already existing functionality. A transformed object is used if a view on multiple business objects is required in the representation of a business object (for example, because the using application requires a business object). Transformed objects are instantiated at run-time and do not have their own persistence – they rely on the persistence of the underlying business objects.
Examples of transformed objects:
- Party
- UsedAddress
- SiteLogisticsTaskExecutionView

Dependent Object
A dependent business object is used as a reusable part in another business object. It represents a concept that cannot stand alone from a business point of view. Instances of dependent objects can only occur in the context of other business objects.
Examples of dependent objects:
- Access Control List
- Address
- Attachment Folder
- Tax Calculation
- Text Collection
- Time Based Accrual
- Work Agreement Payroll Input
- Workplace Address

Template Object
A template object is used to describe objects representing similar subject matter. It specifies the union of all components (nodes, relationships, elements, service operations) of the objects, without any redundancy.
Examples of template objects:

- Customer Transaction Document (this is the template for SalesOrder, Quote, Return, ServiceOrder, ServiceRequest, and so on)
- Product (this is the template for Material, Service Product, Warranty, and so on)

3.7 Test Shell Quick Guide

The Test Shell work center view allows you to browse the SAP business objects and entities that you can use in developing your solution. The work center view contains three separate sub-views:

Business Objects
The Business Objects view enables you to browse the business objects that are released in the public solution model (PSM) and the business objects you have created. You can view the business object metadata as well as business object instances.

You can use this view to answer questions such as:

- Which business objects, nodes and elements can I create/delete/update?
- What is the data type of a specific attribute of a business object?
- What code values exist for a specific code list data type?
- Which business objects, nodes and elements are deprecated?
- What associations exist for a specific business object?
- What queries exist for a specific business object?
- What data exists for a business object instance?

Embedded Components
The Embedded Components view enables you to browse the embedded components that are released in the public solution model (PSM).

Object Value Selectors
The Object Value Selector view enables you to browse the object value selectors that are released in the public solution model (PSM).

The Test Shell functionality is available to partners who are developing scalable solutions on a development tenant as well as to partners and resellers who are developing customer-specific solutions on a development tenant or a customer’s tenant. The functionality is not available on a customer’s production tenant after go-live.
For example, you want to know which time zones are defined for a sales order.

1. Open the Business Objects work center view
2. Type SalesOrder in the Find field and click Go.
3. Select the SalesOrder and click Show Node Overview.
4. Enter the text Period in the Find field and click Go.
5. Select the PeriodTerms node and click Show Attribute Overview.
6. Enter the text TimeZone in the Data Type field and click Go.
7. In the Details area, click the TimeZoneCode link.
8. On the Data Type: TimeZoneCode tab, click Go.

You can now see a list of the time zone codes defined for a sales order in SAP Business ByDesign.

Tasks

Display the Metadata for a Business Object

In the Test Shell work center, Business Object view, type a business object name in the Find field and click Go.
You can perform an advanced search by specifying the Deployment Unit, Namespace and Business Object.

The search terms are case sensitive – and SAP business objects are named with CamelCase. For example, SalesOrder, EmployeeTime or ServiceProduct.

For each search result the following information is displayed:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Business Object Namespace | The namespace of the business object. The following namespaces are accessible:  
http://sap.com/xi/AP*  
http://sap.com/xi/Documentservices/Global  
your partner namespace |
| Object Category | The business object category. The following categories are accessible:  
Standard Business Object  
Dependent Object  
Transformed Object  
Pre-Delivered Object |
| Technical Category | The business object technical category. The following technical categories are accessible:  
Business Transaction Document  
Master Data Object  
Business Configuration Object  
Business Administration Object |
## Name Description

### Release Status
The Release Status has three possible values:
- **Released** - represented by a green square
  Entities that are released in the PSM and may be accessed from your partner solution.
- **Deprecated** - represented by a yellow triangle
  Entities that will be discontinued with the next ByD release. If your solution uses a deprecated entity, you need to adjust the solution to remove references to deprecated entities.
- **Partner Entity** - represented by a circle
  Entities that exist in solutions belonging to your partner namespace.

### Write Access for Partners
Indicates whether you can create, update or delete a business object instance.

### Deployment Unit
The deployment unit of the business object.

### Supported Access Contexts for Defining Access Restrictions
The access contexts for which instance-based authorization is supported.

Select a business object and click **Details** to access the following views:

- **Business Object Structure Overview**
  The business object is presented as an expandable tree-structure that allows you to drill-down through the following information:
  - Actions
  - Associations
  - Queries
  - Keys
  - Nodes

- **Node Overview**
- **Node Attribute Overview**
- **Association Overview**
- **Association Filter Parameter Overview**
- **Query Overview**
- **Query Attribute Overview**
- **Action Overview**
- **Action Parameter Overview**

### Display the Instances of a Business Object

1. In the **Test Shell** work center, **Business Object** view, type a business object name in the **Find** field and click **Go**.
2. In the **Business Objects** view, select a business object and click **Show Business Object Instances**.
   You can also launch the **Business Object Instances** view from **Partner Development > Common Tasks > Show Business Object Instances**.
3. In the **Execute Query** dialog box, select the query to run by entering information in the **Business Object Namespace**, **Business Object**, **Node**, and **Query** fields.
4. Modify the query by selecting options in the lower table. For example, you could filter the results to a specific date range or you could exclude certain results from the query.

5. Click OK to execute the query.

6. Select a business object instance from the query results to view detailed information.

7. Click the Previous Attributes and Next Attributes buttons to navigate left and right through the results.

8. Select the Output Formatting Active check box to toggle between the internal and external values for attributes. For example, instead of Status Code = 1 you will see Status Code = Released.

You can search for a business object instance using an Alternative Key by clicking Load BO Instances By Alternative Key.

You can find associated business objects by clicking Navigate By Association.

3.8 Example: Building a Simple Solution for SAP Business ByDesign

3.8.1 Example: Building a Solution for SAP Business ByDesign

Overview

In this example you are going to build solution to manage the allocation of car parking spaces to employees in the SAP Business ByDesign solution.

- The solution contains one business object – representing a car park – of which there can be multiple instances.
- The car park business object contains a node for "parking spaces" – of which there can be 0 to n.
- Each parking space is assigned to an Employee ID using an Object Value Selector (OVS) – this allows you to look-up the employees within the company.
- The parking space node has an association to the Employee business object – this allows you to retrieve the employee’s first and last name.

Any software coding or code lines/strings ("Code") provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

Tasks

Create the Business Object

1. Open the SDK and log on to the repository.
2. In the Repository View, right-click My Solutions and click Create Solution. The Create Solution dialog box opens.
3. Enter a Name and Description for your solution, select Create Sandbox Solution, and click OK.
4. In the Solution Explorer, right-click the project and select Add New Item.
5. Select the Business Object template and enter a Name (for example CarPark) then click Add. Each instance of the business object represents a car park and the company may have several car parks.
To add data to your business object, you declare an “element”, enter a name, and assign it a data type. Elements are entered according to the following syntax:

\[
element <\text{your_element_name}> : <\text{data_type}>;
\]

When you type the colon (:) the code completion function presents a list of valid data types, from which you can select.

6. Add some data to the car park. First you need an ID to identify each individual car park. You can then add extra information, such as a name and the number of car spaces.

\[
\begin{align*}
\text{element CarParkID : ID;} \\
\text{element CarParkName : LANGUAGEINDEPENDENT_Text;} \\
\text{element CarParkDescription : LANGUAGEINDEPENDENT_Text;} \\
\text{element Number_of_ParkingSpaces : IntegerValue;}
\end{align*}
\]

7. Each car park can have multiple car parking spaces. Therefore, you create a node with a multiplicity of \([0..n]\) to represent the car spaces within each car park. Each node requires an ID.

\[
\begin{aligned}
\text{node ParkingSpace [0..n] { } } \\
\quad \text{element ParkingSpaceID : ID;}
\end{aligned}
\]

8. Each car space can be assigned to an employee. Therefore, you create an element of type EmployeeID to record the employee.

\[
\text{element ParkingSpace_AssignedEmployeeID : EmployeeID;}
\]

9. To make the connection to the existing employees in the company, you create an association to the SAP business object: Employee. This enables you to use the data from this business object in your screens. To use the Employee association you first have to import the namespace of the Employee business object.

\[
\begin{align*}
\text{import AP.FO.BusinessPartner.Global;} \\
\text{association ToEmployee [0..1] to Employee;}
\end{align*}
\]

10. You can add some additional information for the car space.

\[
\begin{align*}
\text{element ParkingSpace_Disabled : Indicator;} \\
\text{element Vehicle_RegistrationNumber : LANGUAGEINDEPENDENT_Text;} \\
\text{element ParkingSpace_Level : NumberValue;}
\end{align*}
\]

11. Save and activate the business object.

You can find the complete business object code here.

Create the Screens

The next step is to create the screens that will display the data from the business object, and allow the user to interact with the business object.

1. Right-click your business object and select Create Screens.

2. Enter a Name for your screens and select Floorplan Scenario with Navigation. The default screens provide a starting point for your solution and include built-in navigation between screens. For this example, you will use the default Object Work List (OWL) and you will modify the Quick Activity Floorplan (QAF) screen.

3. In the Solution Explorer, double-click the QA floorplan to open it in the user interface designer (UI designer). On the Designer tab, you can see that the root node elements from the business object are automatically displayed as fields on the screen.
On the DataModel tab you can see the data fields to the left, and on the right you can see the structure of your business object, including the ParkingSpace node and the association to the SAP business object: Employee. Notice that data fields are automatically created for the root node elements. Selecting one of the data fields shows that a binding exists with the corresponding business object element.

Create the Data List

For each car park, you want a table listing the car spaces and the assigned employee for each space.

1. Click the Display <> Edit button to switch to edit mode.
2. Open the Designer tab and click Add FlexLayoutRow ( ).
3. Select the new row.
4. Drag an Advanced List Pane from the Toolbox and drop it onto the new row.
The list pane is displayed on the **Designer** tab.

5. Right-click one of the columns and select **Add Column**.
6. Open the **BO Browser/Data Model** from the **View** menu.
7. Drag the **ParkingSpaceID** element from the **BO Browser** onto the first column header of the list pane. Select **No** when prompted to use backend text.

On the **DataModel** tab you can see that the data field and the binding are created automatically.
8. Drag the `ParkingSpace_AssignedEmployeeID` element onto the second column header of the list pane. Select No when prompted to use backend text.

Add the Object Value Selector

You can use an Object Value Selector (OVS) to select the employee ID from a list of the existing employees.

1. Select the `ParkingSpace_AssignedEmployeeID` column and in the Property Explorer, under Appearance, select `ObjectValueSelector` as the `DisplayType`. 
2. Under **ValueHelp**, set the **OVSComponent** to: \SAP\_BYD\_APPLICATION\_UI/publicovs/employeeid/
   EmployeeID:
   a. In the **OVS Component** field, click ...
      The **Select OVS Component...** window opens.

   ![Select OVS Component](image1)

   b. Click **Browse**.
      The **Repository Browser** opens.

   ![Repository Browser](image2)

   c. Select the required component, then click **OK**.

**Add the Associated Elements**

1. Open the **BO Browser/Data Model** from the **View** menu.

2. In the **BO Browser/Data Model**, expand the **ToEmployee** association. Here you can see the fields of the SAP business object: **Employee**.

3. Select the element **ToEmployee > CurrentCommon > Person > GivenName** and drop it onto the third column header.

4. Select the element **ToEmployee > CurrentCommon > Person > Name > FamilyName** and drop it onto the fourth column header.
   Both the first and last name will be automatically populated based on the selection of the Employee ID, therefore you do not want these fields to be editable.
5. Select each column and set the Properties > Appearance > Display Type to StaticText.

Edit the Column Headers

For each column, edit the header Label in the Property Explorer.

Suggested headings are as follows:

- Parking Space ID
- Assigned Employee ID
- First Name
- Last Name

Proceed as follows:

1. Click the Arrow Down button in the Label field.
   The DependentPropertyEditor… window opens.
2. In the **Overridden Text** field, change the label name and click **OK**.

### Add a Toolbar

1. Select the list pane by clicking the area to the left of the first column.
2. In the **Property Explorer** select **AdvancedListPaneVariant: ListPaneVariant**.

3. Under **Toolbar Information**, set the **UseToolbar** value to **True**.
   By default, you get an **Add Row** button and a **Remove** button. However, you have to add and configure an event handler for each button.
4. Click the **Add Row** button.
5. In the **Property Explorer**, under **Events** select ➤ **OnClick** ➤ **New Event Handler** ➤
   The **Configure Event Handler** dialog box opens.
6. Enter a name for the event handler (for example **AddRowHandler**).
7. Add an operation of type **List**.
8. Set the **Operation** to **AddRow**.
9. Set the **Target List** to **/Root/DataList**.
10. Click **OK**.
11. Repeat the above steps for the **Remove** button, but this time give it the name **RemoveRowHandler** and select **RemoveRow** as the **Operation**.
12. Add an additional operation of type **WindowAction** and select **Save** as the **Action Type**. This ensures that a save is performed after a row is deleted.

### Change the Floorplan Title

You can create a variable title, so that the title reads "Car Park X" – where "X" is the car park ID.

1. On the **Designer** tab, select the **Identification Region**.
2. In the **Property Explorer**, under **Misc** select **Floorplan Title**.
3. In the **Dependent Property Editor**, select **Overridden Text** and click **Advanced**.
4. Replace the existing **Constant/DataField** with the text "**Car Park:** “, including the blank space at the end.
5. Click the plus button to create a new string or data field.
6. Click the three dots at the end of the new row.
7. In the **Select Data Field** dialog box, select the data field **CarParkID**.
8. Click **OK**.

```
Car Park: {0}
```

### Change the Title of the List Pane

1. On the **Designer** tab, select the list pane.
2. In the **Property Explorer**, under **Pane Header** select **Pane Header**.
3. In the **Dependent Property Editor**, select **Overridden Text** and enter a name for the pane (for example **Parking Spaces**).
Remove the Title of the Form Pane

1. On the Designer tab, click the text Root Node Elements to select the Section Group within the form pane.
2. In the Property Explorer, under Text Information set Show Header to False.

The title of the form pane disappears.

Change the Order of the Fields in the Form Pane

1. Select the Section Group containing the fields CarParkID and so on.
2. In the Property Explorer, select Fields under Contents.
   The Field Collection Editor dialog box is displayed.
3. Select a field and click the arrows to move it up or down the order.

Edit the Field Labels in the Form Pane

To edit the name of a field, select the field and edit the Label in the Property Explorer.

Add Details Area

1. On the Designer tab, select the Parking Spaces section and click Add FlexLayoutRow (/uploads/).
2. Open the BO Browser / Data Model.
3. Drag and drop elements from the BO Browser onto the new row:
   • ParkingSpace_Disabled
   • ParkingSpace_Level
   • Vehicle_RegistrationNumber
   • ElectricChargeStation
4. For each field, edit the Label in the Property Explorer (Disabled Friendly Parking, Parking Level, Vehicle Registration Number).
5. Select the Section Group and change the Field Group Header text in the Property Explorer to Details.
6. Open the **DataModel** tab and drag the newly added fields into the **DataList**. This links these fields to the individual parking spaces.
7. Click **Save and Activate**.
8. Close the UI designer.

### Edit the Work Center Title

The default work center title is taken from the business object name. You can change this title by editing the work center floorplan (WCF).

1. In the **Solution Explorer**, double-click the WCF screen to open it in UI designer.
2. Click the **Display <> Edit** button to switch to edit mode.
3. On the **Designer** tab, click the folder icon to select the **OberonCenterStructure**.
4. In the **Property Explorer**, click the **Title** field.
5. In the **Dependent Property Editor**, select **Overridden Text** and enter a name for the work center (for example **Car Park Manager**).

   If other users have completed this example, there may be multiple work centers with the same title. In this case, you may want to give the work center a unique title, such as `<your name> – Car Park Manager`.

6. Click **OK**.
7. Click Save and Activate.
8. Close the UI designer.

Create the AfterModify Script File

You have set up the QAF screen and created the OVS to find the Employee ID. However, you need a script file to assign the value selected in the OVS to the Parkingspace_AssignedEmployeeID element and to update the other fields (first name, last name, etc.) after this selection.

1. In the Solution Explorer, right-click the business object and select Create Script Files.
2. In the Create Script Files dialog box, create an AfterModify script file for the ParkingSpace node.
3. Enter the script code from here.
4. Save and activate the script file.

Assign Work Center

Assign rights to your business user so that you can access the new work center in SAP Business ByDesign. For more information, see User Setup Quick Guide [page 39].
Test the Solution

1. Log on to SAP Business ByDesign with a user that has access to the Car Park Manager work center.
2. Click the Car Park Manager work center to open the Car Park OWL.
From the OWL you can create new car parks or edit the details of existing car parks.

3. Select a car park and click **Edit** to bring up the QAF.

4. Click **Add Row** to create a new parking space.
5. Enter a **Parking Space ID**.
6. Click the icon in the **Employee ID** field to bring up the OVS and select an employee. The **First Name** and **Last Name** fields are automatically populated.
7. Enter additional information in the **Details** area.

**Further Development**

Now that you have developed a solution, you can begin to appreciate the power and flexibility of the SDK. There are many enhancements you could make to this solution. Here are just a few ideas:
Auto-refresh the OWL after creating a new car park. For more information, see here [page 102].

Validate the car park on save, so that you cannot assign more than the total number of car spaces. An error message is displayed to the user if they attempt to save a car park that fails this validation rule. For more information, see here [page 118].

Create a read-only car park factsheet. For more information, see here [page 103].
  - Create a View button on the OWL that links to the car park factsheet using object-based navigation (OBN). For more information, see here [page 108].
  - Create a link from the car park factsheet to the employee factsheet using object-based navigation (OBN). For more information, see here [page 112].

Extend the employee business object to show the Parking Space ID on the employee factsheet. For more information, see here [page 116].

Add a URL mashup to display the car park location on a map. For more information, see here [page 120].

Add a form to provide a PDF printout of parking spaces and assigned employees.

Add date/time information to each parking space. You could then add a cost per day or per hour.

3.8.2 Further Development

3.8.2.1 Configure the OWL to Auto-Refresh

Overview

In this example you are going to configure the car park object work list (OWL) to refresh automatically when a new car park instance is saved on the quick activity floorplan (QAF).

Prerequisites

- You have completed the example: Building a Solution [page 88]

Procedure

1. Open the OWL in the UI designer.
2. Switch to edit mode.
3. On the Controller tab, create a new Inport with the name AutoRefresh_Inport.
4. In the Property Explorer, perform the following actions:
   - set the property Events OnFire to the default EventHandler – this event handler executes the query for the OWL
   - enter the name AutoRefresh_LocalEvent in the field Navigation Local Event
5. Save and activate and close the UI designer.
6. Open the QAF in the UI designer.
7. Switch to edit mode.
8. On the Controller tab, create a new Outport with the name AutoRefresh_Outport.
9. In the Property Explorer, perform the following actions:
• set the property Behavior > Broadcast to True.
• enter the name AutoRefresh_LocalEvent in the field Navigation > Local Event.

10. On the Controller tab, edit the SaveHandler to include a new operation of type FireOutport below the existing WindowAction operation.

   Operations are execute from top to bottom, so you need to save the new business object instance before running the default query.

11. Save and activate

Result

When you save a new car park instance on the QAF, then return to the OWL, the list of car parks is updated with the car park that you created - you do not have to click Go to run the query again.

3.8.2.2 Create a Factsheet for the Car Park

Overview

In this example you are going to create a factsheet for a car park instance. The factsheet (FS) provides a read-only version of the information entered on the quick activity (QA) floorplan. The main steps in creating a factsheet are as follows:

• create the floorplan layout
• configure the floorplan to allow for object-based navigation from other floorplans to this floorplan:
  ○ add the fields OBN_NAV_KEY and NodeID to the data model
  ○ create the inport
  ○ create the event handler for the inport
• include the FS in the assigned objects of the work center view (WCV)

   It is intended that the factsheet will be accessed from a View button on the object work list (OWL). For more information about creating the button and linking to the car park factsheet, see here [page 108].

Prerequisites

• You have completed the example: Building a Solution [page 88]

Procedure

1. Right-click the CarPark business object and select Create Screens.
2. Select the Factsheet floorplan and click OK.
3. Double-click the factsheet to open it in UI designer.
4. Switch to edit mode.
5. Delete the column on the right.
6. Delete the fields Exldr, Exldr1 and Exldr2.
7. Delete the *Edit* button.
8. Delete the *View All* button.
9. Select the *IdentificationRegion* and edit the *FloorplanTitle* to include the Car Park ID as a variable.
10. Select the *SectionGroup* and set the property *ShowHeader* to false.
11. Select the *SectionGroup* and select the property *Contents > Fields* to open the *Field Collection Editor* and re-order the fields.
12. Select each field and set the property *DisplayType* to *StaticText*.
13. Select each field and edit the *Label* property.
14. Add a flex layout row below the existing row.
15. From the *Toolbox*, drag-and-drop an *AdvancedListPane* into the new row.
16. Edit the property *PaneHeader* to read *Car Spaces*.
17. Add a column to the table.
18. Drag-and-drop the following fields from the *BO Browser/Data Model* onto the table columns:
   - *ParkingSpaceID*
   - *ParkingSpace_AssignedEmployeeID*
   - *GivenName*
   - *FamilyName*
19. Rename the column header labels:
   - *Parking Space ID*
   - *Assigned Employee ID*
   - *First Name*
   - *Last Name*
20. Set the *DisplayType* of the *Assigned Employee ID* column to *Link*, and set the other columns to *StaticText*.
21. Select the *AdvancedListPane: ListPaneVariant* and set the property *UsePreviewPane* to *True*.
22. Edit the property *PaneHeader* to read *Details*.
23. Drag-and-drop the following fields from the *BO Browser/Data Model* into the *Details* section:
   - *ParkingSpace_Level*
   - *Vehicle_RegistrationNumber*
   - *ParkingSpace_Disabled*
   - *ParkingSpace_ElectricChargeStation*
24. Rename the field labels:
   - *Parking Space Level*
   - *Vehicle Registration Number*
   - *Disabled Friendly*
   - *Electric Recharge Station*
25. Change the order the fields as desired.
26. Optional: On the *DataModel* tab, you can add a data structure element to contain the CarPark root node elements. This makes it easier to view and organize the data model elements, but does not make any functional difference.
27. On the **DataModel** tab, perform the following actions:
   a. Right-click the **Root** data field and select **Add Data Field**.
   b. Right-click the new field, select **Rename** and enter the name **OBN_NAV_KEY**.
   c. Right-click the **Root** data field and select **Add Data Field**.
   d. Select the new data field and select the **NodeID** in the business object model, then click **Bind**.
28. On the **Controller** tab, perform the following actions to create the inport:
   
   a. Right-click **Imports** and select **Add Import**.
   
   b. Under **Import Configuration**, select the check box **OBN Inport**.
   
   c. Check that the **Namespace** and **BO Model** fields are populated correctly.
   
   d. In the field **Select Operation**, enter the name **Open_CarPark_Factsheet**. This name is used to identify the inport.
   
   e. Click the three dots next to the **PortType Package** field and select the port type package for the business object. This is located in the folder `<project name>_MAIN/SRC`.
   
   f. In the **PortType Reference** field, select **To_CarPark_Edit**.
   
   g. Click the **Add Parameter** button.
   
   h. Click the three dots in the new **Parameter** row.
   
   i. In the **Parameter Binding** dialog box, select the field **OBN_NAV_KEY** and click **OK**.
j. In the *Property Explorer*, under *Events*, in the *OnFire* field select *New Event Handler*.

k. In the *Configure Event Handler* dialog box, perform the following actions:

   a. Enter the name *Open_CarPark_Factsheet_Handler*.
   b. click the *Add Operation* button.
   c. Select *BOOperation* for the operation *Type*.
   d. Select *Read* for the *BO Operation Type*.
   e. Click the *Add Parameter* button.
   f. For the new parameter row, select *nodeId* in the *Type* field.
   g. Click the three dots next to the *Bind* field.
   h. In the *Parameter Binding* dialog box, select the field *OBN_NAV_KEY* and click *OK*.
   i. Click the three dots next to the *Path* field.
   j. In the *Parameter Binding* dialog box, select the field *NodeID* and click *OK*. 
29. Click **Save and Activate**.
30. In the **Solution Explorer**, double-click the work center view (WCV) to open it in UI designer.
31. Switch to edit mode.
32. In the **Property Explorer**, under **RBAMData** click **AssignedObjects**.
33. In the **Add Assigned Objects** dialog box, click the **Add** button to create a new row.
34. In the new row, click the three dots next to the **ComponentId** field.
35. In the **Select Component** dialog box, navigate to your project and select the factsheet then click **OK**.
36. Click **Save and Activate**.

### 3.8.2.3 Create an OBN to the Car Park Factsheet

**Overview**

In this example you are going to create an object-based navigation (OBN) link from the Car Park Manager object work list (OWL) to the Car Park factsheet. You have already configured the factsheet with an inport that allows for OBN, now you are going to create an OBN outport to connect to this inport.

**Prerequisites**

- You have completed the example: **Building a Solution** [page 88]
- You have completed the example: **Create a Factsheet for a Car Park** [page 103]
Procedure

1. Open the Car Park OWL in UI designer.
2. On the Controller tab, right-click OBN Navigation and select Add OBN. An outport is also created when you create the OBN.
3. Right-click the OBN and rename it to OBN_to_CarPark_Factsheet.
4. Check that the namespace and BO model are correct for your Car Park solution.
5. Click the CarPark node in the BO model and select Open_CarPark_Factsheet in the field Select Operation. The PortType Package and PortType Reference fields are populated automatically.
6. Right-click the outport that was created when you created the OBN and rename it to `Outport_to_CarPark_Factsheet`.
   You will notice that the `PortType Package` and `PortType Reference` fields are pre-populated.
7. Click the `Add Parameter` button.
8. In the parameter row, click the three dots next to `ParameterBinding`.
9. In the `Parameter Binding` dialog box, select the `NodeID` data field within the data list and click `OK`.
10. Right-click `EventHandlers` and select `Add Event Handler`.
11. Right-click the new handler and rename it to `OBN_to_CarPark_Factsheet_Handler`.
12. Click the `Add Operation` button.
13. Select the `Type` as `FireOutport` and select the `Outport` as `Outport_to_CarPark_Factsheet`.

14. Click the `Designer` tab and select the `ListPaneToolbar`.
15. In the `Property Explorer` select to configure the `Toolbar`.
16. In the `Configure Toolbar` dialog box, move the `View` button from `Available` to `Current` and click OK.
17. Select the `View` button and in the `Property Explorer` set the `Events > OnClick` field to `OBN_to_CarPark_Factsheet_Handler`. 
3.8.2.4 Create an OBN to the Employee Factsheet

Overview
In this example you are going to create an object-based navigation (OBN) link from the Employee ID field on the Car Park factsheet to the Employee’s factsheet.

Prerequisites
- You have completed the example: Building a Solution [page 88]
- You have completed the example: Create a Factsheet for the Car Park [page 103]
- You have completed the example: Create an OBN to the Car Park Factsheet [page 108]

Procedure
1. Open the Car Park factsheet in UI designer.
2. On the DataModel tab, add a new data field to the DataList.
3. Select the new data field, select the BO element SAP_ToEmployee and click the Bind button.

18. Click Save and Activate.
4. On the Controller tab, right-click OBN Navigation and select Add OBN. An outport is also created when you create the OBN.

5. Right-click the OBN and rename it to OBN_to_Employee_Factsheet.


7. Select Employee for the BO model.

8. Select Open in the field Select Operation. The PortType Package and PortType Reference fields are populated automatically.
9. Right-click the outport that was created when you created the OBN and rename it to **Outport_to_Employee_Factsheet**.
   You will notice that the **PortType Package** and **PortType Reference** fields are pre-populated.

10. In the **KeyList** row, click the three dots next to **ParameterBinding**.

11. In the **Parameter Binding** dialog box, select the **DataList** and click **OK**.

12. In the **Key** row, click the three dots next to **ParameterBinding**.

13. In the **Parameter Binding** dialog box, select the **SAP_ToEmployee** data field within the data list and click **OK**.
14. Right-click EventHandlers and select Add Event Handler.

15. Right-click the new handler and rename it to Open_Employee_Factsheet_Handler.

16. Click the Add Operation button.

17. Select the Type as FireOutport and select the Outport as Outport_to_Employee_Factsheet.

18. Click the Designer tab and select the Assigned Employee ID column.

19. In the Property Explorer set the Events > OnClick field to Open_Employee_Factsheet_Handler.
21. Click Save and Activate.

3.8.2.5 Extend the Employee Business Object

Overview
In this exercise you will extend the Employee business object and then enhance the Employee factsheet to display the car parking space assigned to the employee. This involves the following steps:
- create business object extension for the Employee business object
- add an extension field to the business object extension
- enhance the Employee factsheet
- add the extension field to the factsheet
- edit the AfterModify script to update the extension field

Prerequisites
- You have completed the example: Building a Solution [page 88]

Procedure
1. In the Solution Explorer, right-click your project and select Add New Item.
2. Select the Business Object Extension template, enter the name Employee_BO_Extension, and click Add.
3. In the **Business Object Selection** dialog box, select the namespace **AP.PO.BusinessPartner.Global** and the business object **Employee**.

4. Open the business object extension and add the following code to the **Common** node:

   ```
   node Common {

   [Tooltip ("If the employee has been assigned a car parking space, the ID is displayed here.")]
   [Label ("Car Parking Space")]
   element Assigned_Car_Parking_Space_ID : Text;

   }
   ```

5. Save and activate the XBO.

6. In the **Solution Explorer**, right-click the **Employee_BO_Extension** and select **Enhance Screen**.

7. Select the floorplan `employeecontactdataqafeco` and click **OK**.

   The floorplan is opened in UI designer.

8. On the **Designer** tab, select the **Basic Information** section group.

9. In the **Extensibility Explorer**, click the **Add Extension Field to Section Group** button.

10. In the **Add Extension Field** dialog box, select the extension field to add to the floorplan and click **Apply**.

    Remember to select the **Is Display Only** check box so that the field will appear as static text.

11. In the **Solution Explorer**, open the **Event-AfterModify** script for the Parking Space node.

12. Add the following code at the end of the script file, just before the closing bracket:

    ```
    foreach (var common in employee.Common)
    {common.Assigned_Car_Parking_Space_ID = this.ParkingSpaceID;}
    ```

    When a car space is modified, this script will update the extension field (`Assigned_Car_Parking_Space_ID`) with the ID of the Parking Space.

13. Save and activate the script file.

### Result

For employee’s that have an assigned car parking space, the ID of the parking space is displayed on the employee’s factsheet.
Because the value gets assigned in the AfterModify script for the Parking Space node, it will not affect existing entries. You need to create a new entry in the car park to see the result on the Employee factsheet.

3.8.2.6 Validate On Save

Overview

In this example you are going to create a validation script that will check if there are any car spaces available. The script checks if the number of parking spaces assigned is greater than the total number of parking spaces. If there are no available spaces, an error message is displayed and you will not be able to save your changes to the car park.

Prerequisites

- You have completed the example: Building a Solution  [page 88]

Procedure

1. Edit the car park business object to declare the message and specify the message text.
   a. Declare the message after the business object name:
      ```java
      businessobject CarPark_Manager raises Message_Car_Park_Full {
      ```
   b. Enter the message text at the root node level:
message Message_Car_Park_Full text "The car park is full - no spaces available";

2. Save and activate.

3. Right-click the business object and select Create Script Files.

4. Create the Validation: On Save script file for the root node.

5. In the Validation: On Save script file, enter the following code:

```java
if (this.ParkingSpace.Count() > this.Number_of_ParkingSpaces)
{
    raise Message_Car_Park_Full.Create("E");
    return false;
}
return true;
```

6. Save and activate.

Result

When you edit a car park and create more car spaces than the total number of car spaces, an error message is displayed and you will not be able to save your changes.
3.8.2.7 Add a URL Mashup to an SAP Floorplan

Overview

In this example you are going to add a URL mashup to the quick activity (QA) floorplan. This mashup will display the car park location on a map, based on the location information entered.
Prerequisites

- You have completed the example: Building a Solution [page 88]
- The mashup has been created in the SAP solution. For more information, see Mashups Quick Guide [page 368].

Procedure

1. Add an element on the root level of the Car Park business object to record location information.
   \[\text{element CarParkLocation : LANGUAGEINDEPENDENT\_Text;}\]
2. Save and activate the BO.
3. Double-click the QA to open it in UI designer.
4. Click the Display<>Edit button to switch to edit mode.
5. Click the Update Metadata button to update the BO metadata.
6. Drag the CarParkLocation from the BO Browser / Data Model onto the Designer tab.
7. Change the label for the field to Car Park Location.
8. On the Controller tab, perform the following actions:
   a. Add a new outport with the name URL\_Mashup\_OutPort and the following settings:
      - PortType Package = /SAP\_BYD\_TF/Mashups/globalmashupsupporttypes
      - PortType Reference = Address\_Info
   b. Bind the Address parameter to the CarParkLocation data field.
   c. In the Property Explorer, select the QA floorplan from the drop-down list at the top.
   d. Click the Anchor field under Extensibility.
      The Anchor Modelling dialog box appears.
   e. Click the plus sign to create a stable anchor for the floorplan.
f. In the **Property Explorer**, select the mashup outport from the drop-down list at the top.

g. Click the **Anchor** field under **Extensibility**.
   The **Anchor Modelling** dialog box appears.

h. Click the plus sign to create a stable anchor for the floorplan.

i. In the **Property Explorer**, click the **Referenced Anchor** field under **Extensibility**.
   The **Anchor Modelling** dialog box appears.

j. Click the plus sign to create a referenced anchor for the outport.

k. Click the three dots next to the **XRep Path** field.

l. In the **Configured Anchors** dialog box, click the anchor you created for the QA floorplan.
9. Save and activate.

10. Click the **Designer** tab.

11. Click the **Display<>Edit** button to switch to display mode.

12. In the **Extensibility Explorer**, select the mashup outport and click the **Mashups Management** button.
   
   The **Mashups Management** dialog box appears, listing the available mashups.

13. Select the mashup you want to use and click **Apply**.
14. Save and activate.

**Result**

The mashup is available from the *Web Services* menu on the QA. When you click the mashup, the URL is opened with the value entered into the *Car Park Location* field as an input parameter.
3.9 Using the Localization Toolkit

Overview

The localization toolkit is a set of development tools and instructions that enable SAP partners to extend and adapt the capabilities of the SAP Business ByDesign solution to provide a locally-compliant solution for customers in countries where a fully-localized SAP country version is not available.

The toolkit enables customers and subsidiaries to implement business processes for a number of localization topics in these countries, without the need for the implementation of a fully-localized country version. In this way, it is only necessary to implement the legal and local regulations relevant particular to the business processes required by the customer or subsidiary rather than all country requirements.

The toolkit comprises tools integrated in the SAP Business ByDesign solution, implemented using the SAP Business ByDesign studio, or available as extensibility features such as extension fields. Using the toolkit, you can extend and adapt existing capabilities, for example, you can implement additional business configuration, such as accounting principles and chart of accounts for a new country, create tax content for non-localized countries, extend existing fields, create new reports, or implement different form templates for output documents.

Prerequisites

You have downloaded the SAP Business ByDesign studio from the Download Center in the SAP Service Market Place. For information about the installation steps, the latest builds, and the SAP Service Market Place, see SAP Business ByDesign Installation Corner.
Accessing the Toolkit

The following topics are covered in the localization toolkit. You can access detailed information about these topics in the Business Center at https://www.sme.sap.com under SAP Business ByDesign > Community > Wiki Info Exchange > Country Information > Customer Specific Localization > Localization Toolkit.

- General Ledger
- Tax
- Tax Reporting
- Fixed Assets
- Inventory
- Accounts Payable/Accounts Receivable
- Legal Reporting/Fiscal Closing
- Invoicing
- Payments
- Internationalization
- Languages
- Extensibility Features
4 Administration and Lifecycle Management

4.1 Overview of Administration and Lifecycle Management

Administration and Lifecycle Management encompasses the following reference material:

<table>
<thead>
<tr>
<th>Get Informed About Administration and Lifecycle Management of Customer-Specific Solutions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Administration Quick Guide for Customer-Specific Solutions [page 144]</td>
<td>• Create a Customer-Specific Solution [page 152] Learn how to create solutions for specific customers.</td>
</tr>
<tr>
<td>The Administration toolbar provides access to a number of administrative tasks, such as creating a new solution, deleting a solution, and managing user sessions.</td>
<td>• Create a Solution Template [page 151] Learn about how to create templates for customer-specific solutions.</td>
</tr>
<tr>
<td>• Implementation Manager Quick Guide for Customer-Specific Solutions [page 146] You use the Implementation Manager to manage the lifecycle of customer-specific solutions.</td>
<td>• Patches for Customer-Specific Solutions [page 153] If you need to make changes or corrections to a customer-specific solution after you have assembled it, you can create a patch. Depending on the tenant on which you are working, the process of creating a patch is different.</td>
</tr>
<tr>
<td>• Lifecycle Management of Customer-Specific Solutions [page 127] The SDK supports the full end-to-end lifecycle management of customer-specific solutions. This includes developing, testing, and deploying your solution, as well as creating patches to deliver updates to your customer.</td>
<td></td>
</tr>
</tbody>
</table>

4.2 Lifecycle Management of Customer-Specific Solutions

Overview

The SDK supports the full end-to-end lifecycle management of customer-specific solutions. This includes developing, testing, and deploying your solution as well as creating patches to deliver updates to your customer.

There are different scenarios in which you may want to develop a customer-specific solution:

- **Developing on a Customer’s Preproduction Tenant**
  If the customer has not yet gone live with the SAP solution, then you can develop, test, and deploy your customer-specific solution on the customer’s preproduction tenant. In this scenario, you create the solution and activate it on a single tenant. For more information, see Lifecycle Management on a Preproduction Tenant [page 132].

- **Developing on a Customer’s Test Tenant**
  If the customer has already gone live with the SAP solution, then you develop and test your customer-specific solution on the customer’s test tenant. In this case, development is done on the test tenant and the finalized solution is uploaded and activated on the customer’s production tenant. For more information, see Lifecycle Management on a Test Tenant [page 135].

- **Developing on Your Development Tenant**
  You develop your customer-specific solution on your development tenant in your company or organization. On this tenant you can develop solutions for different customers. The finalized solution is then downloaded
from your development tenant and uploaded and activated on the customer’s production tenant. For more information, see Lifecycle Management on Your Development Tenant [page 139].

The following guidelines and restrictions apply to the development of customer-specific solutions in general:

- We recommend that you develop one solution per customer.
- The tenant where you develop your solution and the tenant where you upload your solution must be on the same release.

Tenant-Specific Activities in the SDK

The following table shows which activities you can perform on which tenant type.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Development Tenant</th>
<th>Customer’s Test Tenant</th>
<th>Customer’s Preproduction Tenant</th>
<th>For Information, See ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Solution</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Create a Customer-Specific Solution [page 152]</td>
</tr>
<tr>
<td>Create Copy of Solution</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Implementation Manager Quick Guide [page 146]</td>
</tr>
<tr>
<td>Create Patch</td>
<td>Created in same namespace</td>
<td>Created in separate namespace</td>
<td>Created in same namespace</td>
<td>Patches for Customer-Specific Solutions [page 153]</td>
</tr>
<tr>
<td>Create Solution Template</td>
<td>X</td>
<td>X</td>
<td>–</td>
<td>Solution Templates Quick Guide [page 149]</td>
</tr>
<tr>
<td>Create Copy of Template</td>
<td>X</td>
<td>X</td>
<td>–</td>
<td>Solution Templates Quick Guide [page 149]</td>
</tr>
<tr>
<td>Switch Customer</td>
<td>X</td>
<td>–</td>
<td>–</td>
<td>Switch a Customer Assignment [page 143]</td>
</tr>
<tr>
<td>Create and Maintain Solution With BC Sets Using SAP BCOs</td>
<td>X</td>
<td>–</td>
<td>–</td>
<td>Create a BC Set Using an SAP BCO [page 290]</td>
</tr>
</tbody>
</table>

See Also

Maintenance Mode [page 154]

4.3 Quality Criteria for Customer-Specific Solutions

Overview

Customer-specific solutions for the SAP solution are installed in a multi-tenant environment. The solution provider needs to ensure that all solutions are of the highest quality and have the lowest possible Total Cost of Ownership (TCO).

SAP requests you to conform to the criteria described in this document in order to ensure the quality of your solution.
Functional Correctness
You must ensure that your solution has been tested and documentation is provided for all test cases.

Performance
You must ensure that your solution does not adversely impact the performance of the SAP solution.

Response Time
Where the add-on extends an existing user interface, a performance degradation of maximum 10% is acceptable even if the response time exceeds the target value.

Simple Transactions
The end-user response time for simple transactions must be:

- On average end-to-end < 1 s
- 95% end-to-end < 1.5 s

Examples of simple transactions are: Object Work Lists, purchase request, leave request.

Complex Transactions
The end-user response time for complex transactions must be:

- On average end-to-end < 1.7 s
- 95% end-to-end < 2.5 s

Examples of complex transactions are: integrated transactions like order, but also simple analytical content and fact sheets.

Complex Tasks
To enhance the end-user experience for complex tasks, the following must be true:

- Progress indicator available
- Background processing possible

Examples of complex tasks are: analytical content, planning, multilevel bill of material explosion.

Security
You must implement measures to ensure the security of your solution. Furthermore, your solution must not impact the security of the SAP solution.

Code Scan
Security code scans are an important measure to minimize security risks. For customer-specific solutions, no code scans are performed by default. SAP reserves the right to perform code scans of customer-specific solutions in order to detect possible security risks.

If your solution functionality runs on an external system, you must conduct a security code scan or assign a third party to conduct a security code scan of the external component. SAP reserves the right to request proof of a successful code scan.

As there is no default security code scan for customer-specific solutions, you must take action to avoid the top 10 security risks as defined by the Open Web Application Security Project:

Preventative Measures

You must implement measures to prevent security attacks. You can find more information on preventing common security attacks at the following links:

- Cross-site scripting attacks (http://www.owasp.org/index.php/Cross-site_Scripting_(XSS))
- SQL injection attacks (http://www.owasp.org/index.php/SQL_Injection)
- Directory traversal attacks (http://www.owasp.org/index.php/Path_Traversal)

Useful guidelines and hints on secure programming can be found in the SAP NetWeaver Developer’s Guide: http://help.sap.com/saphelp_nw2004s/helpdata/en/58/4d767ed850443c891ad27208789f56/frameset.htm

Backdoors

The solution must be free of intentionally hidden and non-documented features that circumvent security measures (backdoors).

Vulnerabilities

The solution must prevent typical vulnerabilities such as:

- Missing input validation
- Execution of commands that include not-validated user input
- Parameter tampering
- Cross-site request forgery
- Information leaks (for example, in error messages or hidden HTML fields)

Information on Web-application security can be found here:


Documentation

Documentation must be provided to describe the following:

- Authorization and roles
- Communication security (including port information)
- User management and authentication
- Secure configuration of the service
- Accessing the security logs
- Configuring authentication policies (for example, behavior after a defined number of failed login attempts)
- Restricting access to specific IP addresses

You must provide customers with updates to the security documentation (for example, security bulletins) when applicable.

Authorizations

The solution must provide the capability to separate the authorizations (roles) for different types of users (that is, administrative and non-administrative users). Users must only have the permissions that are required for their work...
(segregation of duties). The authorization concept must be documented (that is, delivered roles per business and technical user). Search results may contain sensitive data and therefore must be filtered based on the authorization level of the user.

**Data Transfer**
The transfer of data to other services or service providers must be configurable and documented.

**Testing**
Details must be provided of any security-related test activities performed.

**Security Settings**
Security settings and recommendations must be documented and available to the customer.

**Supportability**
You must provide for the maintenance and support of your partner solution throughout its entire product lifecycle.

**Maintenance and Support – Customer**

**Customer Service Level Agreement (SLA)**
As you are fully responsible for the solution, you have a support contract with your customer. You must describe any Customer SLA or Customer Support Agreement that exists for your partner solution.

**Maintenance**
Your solution must have a defined release and maintenance strategy and process, which must be linked with SAP’s interface, product, and maintenance process for the SAP solution. This is required as SAP reserves the right to deprecate entities or functions in the SAP solution with a new release, which might have an impact on your solution and might require that you need to update your solution.

**Maintenance and Support – SAP**

**Support Process**
Support processes/interaction between your support department and SAP Support must be aligned and documented.

**Interface Deprecation**
You must be able to react, properly and in time, to the possible deprecation of interfaces from SAP’s public solution model (PSM).

Interfaces of the PSM can be deprecated by SAP. Generally, SAP provides alternative interfaces that continue to offer the required business functionality.

SAP promises that deprecated interfaces remain fully functional within two releases (approximately one year). During this period of time, you need to react by removing usage of the deprecated interfaces from your partner solution.

**Public Solution Model Access**
You must only access SAP interfaces according to the PSM.

**Switching the Solution On/Off**
The SAP solution must remain fully functional if a customer-specific solution is switched off. If your solution jeopardizes the standard functionality of the SAP solution, SAP reserves the right to switch off your solution.

If your solution is switched on again, it then must be able to resynchronize to the existing data in the system.
4.4 Developing on a Customer’s Tenant

4.4.1 Lifecycle Management for Customer-Specific Solutions on a Customer’s Preproduction Tenant

Overview

If the customer has not yet gone live with the SAP solution, then you can develop, test, and deploy your customer-specific solution on the customer’s preproduction tenant. In this scenario, you create the solution and activate it on a single tenant.

The following guidelines and restrictions apply if you work on a preproduction tenant:

- The customer can go live with SAP’s on-demand solution, if there is no customer-specific solution on the tenant that has the status In Development.
- If you create data for testing purposes on a preproduction tenant, be aware that this data will remain on the tenant even after the customer has gone live.
- If the customer has gone live, additional test tenants that had been requested for the preproduction tenant are not available anymore. Therefore you must assemble and download all solutions that are still in development on these test tenants before the customer goes live.
- If you are developing on a test tenant, which the customer has requested for a change project, be aware that the test tenant is not available anymore after the customer has merged the changes back to the production system. Assemble and download your solutions that are still in development before the changes are merged.

The following graphic gives a schematic overview of the lifecycle management process. The process is explained in detail below.
Prerequisites

In the SAP solution, your customer has set up the following users for you:

- A business user for testing purposes in the SAP solution on the preproduction tenant.
- Users with the roles **SDK Administrator** and **SDK Developer** on the preproduction tenant. If you both develop and implement the solution, you only need a user with the **SDK Administrator** role. For more information, see [User Setup Quick Guide](#) [page 39] and [User Roles](#) [page 38].

Process Flow

1. In the SDK, you log on to the preproduction tenant and create a solution (**SDK Administrator**). For more information, see [Create a Customer-Specific Solution](#) [page 152].

2. You develop your solution including the necessary business configuration content (**SDK Developer**). For more information about how to develop different entities for a solution, see [Developer Desktop](#) [page 260]. For more information about how to create business configuration content, see [Business Configuration Quick Guide](#) [page 272].

   You can only create a BC set using an SAP BCO if you are developing the solution on your development tenant.

3. You test the solution (**Business User**). You can test specific solution content in the SDK using the preview function. You can also test your solution in the SAP solution for the tenant on which you created the solution. As a prerequisite, you need to scope your solution. You have the following options:
   - In the SDK, you trigger deployment of your business configuration content (**SDK Administrator**).

      When you use the **Deploy Business Configuration** function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the **Deploy Business Configuration** function deploys all business configuration content independently of any business options you have created.

      You then log on to the SAP solution with your business user and perform your tests.

      - In the SAP solution, you scope your solution manually (**Business User**). To do this, you log on to the SAP solution with your business user. In the **Business Configuration** work center, you perform scoping for the business option that you created for your solution and for the required standard functions of the SAP solution. You then perform your tests.

4. You assemble and download the solution (**SDK Administrator**). After you have performed your tests successfully, in the SDK you assemble and download your solution. For more information, see the [Implementation Manager Quick Guide for Customer-Specific Solutions](#) [page 146].

   If you have used the **Deploy Business Configuration** function in the previous step, the system now removes the business configuration content from the SAP solution. Your customer then needs to activate your solution on the preproduction tenant by selecting the relevant business option in scoping.
Follow-On Activities

After the solution has been activated on the preproduction tenant, you can only make changes or corrections to the solution by creating a patch. For more information, see Create a Patch on a Customer’s Preproduction Tenant [page 134].

If you have uploaded the solution on another tenant, for example, a test tenant, you must then create the patch in this test tenant. In this case, you cannot create a patch on the preproduction tenant anymore.

4.4.2 Create a Patch on a Customer’s Preproduction Tenant

Overview

If you need to make changes or corrections to a customer-specific solution on the customer’s preproduction tenant, you can create a patch. The patch is then created in the same solution and namespace but with a different version number. For more information about patches, see Patches for Customer-Specific Solutions [page 153].

Prerequisites

In the SDK, you have logged on to the customer’s preproduction tenant where you created the original solution and you have opened this solution.

Only users with the **SDK Administrator** role can create patches.

Procedure

1. On the **Implementation Manager** toolbar, click the **Create Patch** button.
   The system creates a patch in the same solution and sets the solution status to **In Development**. The version number of the solution is updated to the next higher number.
2. Your can now implement the required changes.
   When you have finished making your changes, you can activate and then test your solution.
3. In the **Solution Explorer**, right-click your solution and choose **Activate**.
4. To test your solution, you can do one of the following:
   - In the SDK, trigger deployment of your business configuration content. To do this, in the **Solution Explorer**, right-click the **Business Configuration** node in your solution and select **Deploy Business Configuration**.
     When you use the **Deploy Business Configuration** function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the **Deploy Business Configuration** function deploys all business configuration content independently of any business options you have created.
You then log on to the SAP solution with your business user and perform your tests.

- In the SAP solution, you scope your solution manually.
  To do this, you log on to the SAP solution with your business user. In the Business Configuration work center, you perform scoping for the business option you created for your solution and for the required standard functions of the SAP solution.
  You then perform your tests.

When you have finished testing, you can assemble your solution in the SDK.

5. In the Solution Explorer, right-click the project of your solution and select Check In All Files.
6. On the Implementation Manager toolbar, click the Assemble and Download button.
   The system assembles the patch and downloads it to a .zip file.

   If you have used the Deploy Business Configuration function in the previous step, the system now removes the business configuration content from the SAP solution.

Result

The customer activates your solution in the preproduction system by selecting the relevant business option in scoping.

See Also

Lifecycle Management on a Customer’s Preproduction Tenant [page 132]
Maintenance Mode [page 154]

4.4.3 Lifecycle Management for Customer-Specific Solutions on a Customer’s Test Tenant

Overview

If your customer has already gone live with the SAP solution, you can develop and test your customer-specific solution on the customer’s test tenant. In this case, development is done on the test tenant and the finalized solution is uploaded and activated on the customer’s production tenant. For information about how to request a test system, see the documentation of the SAP solution.

The following guidelines and restrictions apply if you work on a test tenant:

- If the test tenant is a copy of the customer’s production tenant, on which your solution is already activated, you can only create a patch for this solution. For more information, see Create a Patch on a Customer’s Test Tenant [page 137].
- If the test tenant is a tenant that the customer has requested for a preproduction tenant, be aware that the test tenant is not available anymore after the customer has gone live with the respective production system. Assemble and download your solutions that are still in development before the changes are merged.
- If the customer has gone live, additional test tenants that had been requested for the preproduction tenant are not available anymore. Therefore you must assemble and download all solutions that are still in development on these test tenants before the customer goes live.
The following graphic gives a schematic overview of the lifecycle management process. The process is explained in detail below.

Prerequisites

In the SAP solution, your customer has set up the following users for you:

- A business user for testing purposes in the SAP solution on the test tenant.
- Users with the roles **SDK Administrator** and **SDK Developer** in any tenants in which you are going to develop and deploy the solution. If you both develop and implement the solution, you only need a user with the **SDK Administrator** role. For more information, see *User Setup Quick Guide* [page 39] and *User Roles* [page 38].

Process Flow

1. In the SDK, you log on to the customer’s test tenant and create a solution (**SDK Administrator**). For more information, see *Create a Customer-Specific Solution* [page 152].

2. You develop your solution including the necessary business configuration content (**SDK Developer**). For more information about how to develop different entities for a solution, see *Developer Desktop* [page 260]. For more information about how to create business configuration content, see *Business Configuration Quick Guide* [page 272].
   
   You can create BC Sets using SAP BCOs only on a development tenant.

3. You test the solution (**Business User**).
   
   You can test specific solution content in the SDK using the preview function. You can also test your solution in the SAP solution for the tenant on which you created the solution. As a prerequisite, you need to scope your solution. You have the following options:
   
   - In the SDK, you trigger deployment of your business configuration content (**SDK Administrator**).
When you use the Deploy Business Configuration function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the Deploy Business Configuration function deploys all business configuration content independently of any business options you have created.

You then log on to the SAP solution with your business user and perform your tests.

- In the SAP solution, you scope your solution manually (Business User). To do this, you log on to the SAP solution with your business user. In the Business Configuration work center, you perform scoping for the business option that you created for your solution and for the required standard functions of the SAP solution. You then perform your tests.

4. You assemble and download the solution (SDK Administrator).
   After you have performed your tests successfully, in the SDK, you log on to the test tenant to assemble and download your solution. For more information, see the Implementation Manager Quick Guide for Customer-Specific Solutions [page 146].

5. You upload the solution to the production tenant (SDK Administrator).
   In the SDK, you log on to the production tenant and upload the solution.

6. You activate the solution on the production tenant (SDK Administrator).
   After the solution has been successfully uploaded, you activate it.

    Before activating the solution, we recommend that you inform the customer of any functions that may be affected by the update. For example, if the key user at the customer has adapted an analytical report, this may be affected by any changes you have made to the same report.

7. The customer activates your solution by selecting the relevant business option in Scoping.

Follow-On Activities

After the solution has been activated on the production tenant, you can only make changes or corrections to it by creating a patch. A patch follows the same lifecycle as the original solution, with the exception that the business option does not need to be selected again.

4.4.4 Create a Patch on a Customer's Test Tenant

Overview

If you need to make changes or corrections to a customer-specific solution after you have uploaded it to the customer’s production tenant, you can create a patch.

For more information about patches, see Patches for Customer-Specific Solutions [page 153].
The following guidelines and restrictions apply:

- Any data created for the original solution in the SAP solution and any scoping decisions and fine-tuning settings are not copied to the patch solution. For testing purposes, you must therefore scope the patch solution and create new test data.
- If your original solution contains a Web service, this Web service will have a different namespace in the patch solution. You must therefore adapt the settings of your client testing tool to test the Web service in your patch solution.

Prerequisites

In the SDK, you have logged on to the customer’s test tenant where you created the original solution and you have opened this solution.

Only users with the **SDK Administrator** role can create patches.

Procedure

1. On the **Implementation Manager** toolbar, click the **Create Patch** button.
   The system creates a patch solution and copies all files in your solution to a new namespace within the patch solution. The namespace is generated automatically and cannot be changed.
   The patch solution is created in the background and this process may take some time. Click **Refresh** in the **Implementation Manager** tool window to update the solution status.

2. In the **Implementation Manager**, open the patch solution by clicking the **Open** button next to the **Patch Solution** field.
   You can always access the patch solution from the original solution by opening the **Implementation Manager** and, on the **Current Version** tab, clicking the **Open** button next to the **Patch Solution** field.
   You can also navigate to the original solution from the patch solution in the same way.

3. In the patch solution, make your changes.

4. You can test specific solution content in the SDK using the preview function. To test your patch solution in the SAP solution, you have to enable the solution on the test tenant as follows:
   a. In the **Implementation Manager**, click the **Enable** button next to the **Enabled for Business User in On-Demand Solution: No** field.
      The “No” changes to a “Yes” and the patch solution is enabled on the test tenant instead of the original solution.
      If a BC change project is open on the tenant before you enable the patch solution on the test tenant, you will not be able to merge the change project while the patch solution is enabled.
   b. In the SAP solution, log on to the test tenant with your business user.
      You will be able to see your patch solution in place of the original solution.
   c. Complete your testing.

5. On the **Implementation Manager** toolbar, click the **Assemble and Download** button.
   The system assembles the patch using the namespace of the original solution and downloads it to a .zip file.
   This allows you to copy your corrections back to the original solution.
6. In the SDK, log on to the production tenant on which the original solution is active, and click the Upload button on the Implementation Manager toolbar. Select the .zip file containing the patch. The system uploads the .zip file with your corrections and checks for any compatibility issues. The solution has the status In Deployment and the solution version number is updated.

7. In the Implementation Manager, select the solution and click Activate.

\* Before activating the solution, we recommend that you inform the customer of any functions that may be affected by the update.

See Also

- Lifecycle Management on a Customer's Preproduction Tenant [page 132]
- Lifecycle Management on a Customer's Test Tenant [page 135]
- Maintenance Mode [page 154]

4.5 Developing on Your Development Tenant

4.5.1 Lifecycle Management of Customer-Specific Solutions on Your Development Tenant

Overview

You can develop a customer-specific solution on your development tenant and after having tested, assembled, and downloaded the solution, you can upload it to the customer’s tenant.

The following graphic gives a schematic overview of the lifecycle management process. The process is explained in detail below.
Prerequisites

- In SAP Business ByDesign, you have set up the following users:
  - Users with the roles **SDK Administrator** and **SDK Developer** on the tenant on which you want to develop and assemble the solution. For more information, see User Setup Quick Guide [page 39] and User Roles [page 38].
  - User with the role **SDK Administrator** on any tenants to which you want to upload the solution, for example, the customer’s production tenant.
  - A **business user** for testing purposes in SAP Business ByDesign on your development tenant.
- In the SDK, you are logged on to your development tenant.

Process Flow

1. In the SDK, you check the current customer assignment and make sure that the ID of the customer for whom you want to create and develop the solution is displayed (**SDK Administrator**). For more information, see Switch a Customer Assignment [page 143].
2. You create a solution (**SDK Administrator**). For more information, see Create a Customer-Specific Solution [page 152].
3. You develop your solution including the necessary business configuration content (**SDK Developer**). For more information about how to develop different entities for a solution, see Developer Desktop [page 260]. For more information about how to create business configuration content, see Business Configuration Quick Guide [page 272].
4. You test the solution (**Business User**). You can test specific solution content in the SDK using the preview function. You can also test your solution in the SAP solution for the tenant on which you created the solution. As a prerequisite, you need to scope your solution. You have the following options:
   - In the SDK, you trigger deployment of your business configuration content (**SDK Administrator**).
When you use the Deploy Business Configuration function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the Deploy Business Configuration function deploys all business configuration content independently of any business options you have created.

You then log on to the SAP solution with your business user and perform your tests.

- In the SAP solution, you scope your solution manually (Business User).
  To do this, you log on to the SAP solution with your business user. In the Business Configuration work center, you perform scoping for the business option that you created for your solution and for the required standard functions of the SAP solution.
  You then perform your tests.

5. You assemble and download the solution (SDK Administrator).
  For more information, see the Implementation Manager Quick Guide for Customer-Specific Solutions [page 146].

6. After you have performed the tests successfully, you log on to the customer’s production tenant in the SDK to upload the solution. You activate the solution on the production tenant (SDK Administrator).

Before activating the solution, we recommend that you inform the customer of any functions that may be affected by the update. For example, if the key user at the customer has adapted an analytical report, this may be affected by any changes you have made to the same report.

7. In SAP Business ByDesign, the customer activates your solution by selecting the relevant business option in scoping.

Follow-On Activities

After a solution has been activated on the customer’s tenant, you can only make changes or corrections to it by creating a patch. A patch follows the same lifecycle as the original solution, with the exception that the business option does not need to be selected again. For more information, see Create a Patch on Your Development Tenant [page 141].

4.5.2 Create a Patch on Your Development Tenant

Overview

If you need to make changes or corrections to a customer-specific solution after you have uploaded it to the customer’s production tenant, you can create a patch.

When you create a patch of a solution on your development tenant, the patch is created in the same solution and namespace but with a different version number. For more information about patches, see Patches for Customer-Specific Solutions [page 153].

Prerequisites

In the SDK, you have logged on to your development tenant where you created the original solution and you have opened this solution.
Only users with the **SDK Administrator** role can create patches.

**Procedure**

1. On the **Implementation Manager** toolbar, click the **Create Patch** button.
   The system creates a patch in the same solution and sets the solution status to **In Development**. The version number of the solution is updated to the next higher number.

2. You can now implement the required changes.
   When you have finished making your changes, you can activate and then test your solution.

3. In the **Solution Explorer**, right-click your solution and choose **Activate**.

4. To test your solution, you can do one of the following:
   - In the SDK, trigger deployment of your business configuration content. To do this, in the **Solution Explorer**, right-click the **Business Configuration** node in your solution and select **Deploy Business Configuration**.

      > When you use the **Deploy Business Configuration** function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the **Deploy Business Configuration** function deploys all business configuration content independently of any business options you have created.

      You then log on to the SAP solution with your business user and perform your tests.

      - In the SAP solution, you scope your solution manually.
        To do this, you log on to the SAP solution with your business user. In the **Business Configuration** work center, you perform scoping for the business option you created for your solution and for the required standard functions of the SAP solution.
        You then perform your tests.

      When you have finished testing, you can download your solution and then upload it to the customer’s tenant.

5. In the SDK, in the **Solution Explorer**, right-click the project of your solution and select **Check In All Files**.

6. On the **Implementation Manager** toolbar, click the **Assemble and Download** button.
   The system assembles the patch and downloads it to a `.zip` file.

7. In the SDK, log off from your development tenant and then log on to the customer’s tenant where the original solution is active.

8. In the **Implementation Manager**, click the **Upload** button. Select the `.zip` file containing the patch.
   The system uploads the `.zip` file with your corrections and checks for any compatibility issues.

9. In the **Implementation Manager**, click the **Activate** button.
   The patch is activated in the SAP solution.

**See Also**

- [Lifecycle Management on Your Development Tenant](#)
- [Maintenance Mode](#)
4.5.3 Switch a Customer Assignment

**Overview**

When you develop customer-specific solutions on your development tenant, you can create solutions for more than one customer. For example, when you work on a solution for customer A, you can save your solution and then switch to work on the solution for customer B. To do this, you need to switch the customer assignment. After you switch, only the solutions of customer B are listed in the *My Solutions* tool window in the SDK. You can now create new solutions for this specific customer.

**Prerequisites**

In the SDK, you are logged on to your development tenant.

- If you are logged on to your development tenant with a user that has the **SDK Developer** role, you can only switch customer assignments. However, when you want to create new customer assignments, you must log on with a user that has the **SDK Administrator** role.

**Procedure**

1. Select Administration → Switch Customer.
2. In the Switch Customer dialog box, check the details of the current customer assignment. Each solution that you create will be assigned to this customer. If you want to switch the assignment, you can do one of the following:
   - To switch to an existing customer assignment, select a customer from the list of existing customers.
   - To create a new customer assignment, click Create Customer.
     a. In the Create Customer dialog box, enter a customer name and the customer ID that SAP has assigned to the customer.
     i. Ask your customer for the ID. In the Customer ID field of the Create Customer dialog box, always enter a 10 digit ID. If the ID is shorter, fill it up with leading zeros, for example, if the ID is 12345678, enter 0012345678.
     b. Click OK.
     c. In the Switch Customer dialog box, select this customer from the list.
3. In the Switch Customer dialog box, click OK.
   The system logs off and you need to log on again to your development tenant with the same user. In the *My Solutions* window, only the solutions of the current customer are listed. If you have not yet created any solutions for this customer, the *My Solutions* window is empty.

**See Also**

- Create a Customer-Specific Solution [page 152]
- Solution Templates Quick Guide [page 149]
- Lifecycle Management on Your Development Tenant [page 139]
4.6 Administration

4.6.1 Administration Quick Guide for Customer-Specific Solutions

Users with the **SDK Administrator** role can use the **Administration** toolbar to access a number of administrative tasks, such as creating and deleting customer-specific solutions or solution templates and managing user sessions.

If the **Administration** toolbar is not visible in the SDK, you can access it by clicking **View** » **Toolbars** » **Administration**.

**Business and Technical Background**

**Lifecycle Management**

The SDK supports the full end-to-end lifecycle management of customer-specific solutions. This includes developing, testing, and deploying your solution as well as creating patches to deliver updates to your customer.

For more information, see **Lifecycle Management of Customer-Specific Solutions** [page 127].

**Patches**

If you need to make changes or corrections to a customer-specific solution after you have uploaded it to the customer’s production tenant, you can create a patch.

For more information, see **Patches for Customer-Specific Solutions** [page 153].

**Maintenance Mode**

A customer-specific solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is **Assembled**. In both cases, you can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer’s production tenant.

For more information, see **Maintenance Mode** [page 154].

**Tasks**

**Create a Customer-Specific Solution**

You can design and develop customer-specific solutions to enhance the standard SAP solution for a specific customer. In the SDK you can create a solution in which all items that you develop or changes that you make to existing entities are stored.

For more information, see **here** [page 152].

**Create a Solution Template**

You can create templates containing reusable items and upload these items to customer-specific solutions.

For more information, see **here** [page 151].
Create a Patch for a Customer-Specific Solution

If you need to make changes or corrections to a customer-specific solution after you have assembled it, you can create a patch. Depending on the tenant on which you are working, the process of creating a patch is different:

- Create a Patch on a Customer's Tenant  [page 137]
- Create a Patch on Your Development Tenant  [page 141]

Delete Locks Using the Session Administrator

Locks occur when an item, such as a business object, is being edited by another user or from another session. As an administrator, you can delete locks for other users; all other users can only delete their own sessions and corresponding locks.

- Deletion of locks may result in the loss of changes made by the locking user, and should only be carried out when absolutely necessary.
- Locks created by the UI designer can only be deleted by the user that created the lock; an administrator cannot delete these locks.

1. On the Administration toolbar, click Session Administrator. The Session Administrator opens.
2. Click the Refresh button to get a list of users with sessions.
3. Select the user for whom the session should be deleted.
4. Click the Delete button.
5. Make sure that the user whose session was deleted logs off the system. If the user continues working without logging off, locks without session will be created. These locks can only be deleted by the user who created the locks. If you are deleting your own session, you must also log off immediately afterwards to avoid creating locks without session.

Delete a Solution

1. Log on to the repository and open the solution.
2. On the Administration toolbar, click Delete Solution and confirm that you want to delete the solution.

You cannot delete a solution on a production tenant. We recommend you switch the solution off by deselecting it in Scoping. Then report an incident to request assistance from SAP.

Further Tasks

You use the Implementation Manager to manage the lifecycle of customer-specific solutions. The Implementation Manager enables you, for example, to assemble and download a solution, upload and activate a solution, and create a patch for a solution. If you are not able to activate a solution, you can carry out different troubleshooting activities in the Implementation Manager.

For more information, see Implementation Manager Quick Guide for Customer-Specific Solutions  [page 146].
4.6.2 Implementation Manager Quick Guide for Customer-Specific Solutions

You use the **Implementation Manager** to manage the lifecycle of customer-specific solutions. The **Implementation Manager** enables you, for example, to assemble and download a solution, upload and activate a solution, and create a patch for a solution. If you are not able to activate a solution, you can carry out different troubleshooting activities in the **Implementation Manager**.

Only users with the Administrator role can access the **Implementation Manager** tools. For more information see [User Roles](#page-38).

If the **Implementation Manager** is not visible in the SDK, you can access it as follows:

- To enable the **Implementation Manager** tool window, click **View > Other Windows > Implementation Manager**.
  The tool window provides a view of the solution status and implementation log at each stage in the process.
- To enable the **Implementation Manager** toolbar, click **View > Toolbars > Implementation Manager**.
  The toolbar provides buttons that allow you to assemble, upload, and create a patch for your solution.

**Business and Technical Background**

**Lifecycle Management**

The SDK supports the full end-to-end lifecycle management of customer-specific solutions. This includes developing, testing, and deploying your solution as well as creating patches to deliver updates to your customer.

For more information, see [Lifecycle Management of Customer-Specific Solutions](#page-127).

**Patches**

If you need to make changes or corrections to a customer-specific solution after you have uploaded it to the customer’s production tenant, you can create a patch.

For more information, see [Patches for Customer-Specific Solutions](#page-153).

**Maintenance Mode**

A customer-specific solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is **Assembled**. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer’s production tenant.

For more information, see [Maintenance Mode](#page-154).

**Solution Templates**

With a solution template you can organize common development content that you want to reuse for customer-specific solutions. Solution templates enable you to rapidly start the development of customer-specific solutions, for example, for a specific industry.

For more information, see [Solution Templates](#page-150).
Tasks

Assemble and Download a Solution

When you have finished developing your solution, you assemble and download it before uploading it to the customer’s tenant.

For information about how to download a solution template, see Solution Templates Quick Guide [page 149].

1. In the SDK, log on to the tenant where you created the solution, and open the solution.
2. In the Solution Explorer, right-click the solution and select Activate.
3. In the Solution Explorer, right-click the project and select Check In All Files.
4. On the Implementation Manager toolbar, click the Assemble and Download button.
5. Enter a file name to save the template as a .zip file locally.

The system checks the solution for consistency and then assembles and downloads all the items in the solution to a .zip file.

After the solution has been assembled and downloaded, the status changes to Assembled. The solution is now locked for further development and any changes can only be made by creating a patch.

Upload and Activate a Solution

After you have assembled and downloaded your solution, you can upload it to the customer’s tenant.

For information about how to import items of a downloaded template, see Solution Templates Quick Guide [page 149].

1. Log on to the tenant.
2. On the Implementation Manager toolbar, click the Upload button and select the .zip file created in the assemble and download step.
   An upload disclaimer appears.
3. Agree to the terms of the upload disclaimer and click Continue.
   The system unpacks the .zip file into the production tenant.
   During the upload process, the system runs compatibility checks on the solution to ensure the following:
   • The tenant where you develop your solution and the tenant where you upload your solution must be on the same release.
   • SAP does not import a patch to its solution at the same time.
   During the upload process, the status of the solution is In Deployment. If no errors or warnings are found during upload, you can activate the solution.
4. In the Implementation Manager, click the Activate button.
   The status of the solution changes to Deployed.

Create a Patch for a Solution

If you need to make changes or corrections to a customer-specific solution after you have assembled it, you can create a patch. Depending on the tenant on which you are working, the process of creating a patch is different:

- Create a Patch on a Customer’s Tenant [page 137]
- Create a Patch on Your Development Tenant [page 141]
Create a Copy of a Solution

You can use a copy of your solution, for example, to continue developing in order to try out different scenarios without disrupting the original. For this, you can download a copy of an existing solution and then upload this copy to the same tenant or a different tenant.

For information about how to copy a template, see Solution Templates Quick Guide [page 149].

You cannot merge the content of the copy back to the source solution. After having achieved a satisfactory development status of the copy, you need to repeat all changes in the original that you made in the copy.

1. In the SDK, log on to the tenant where you created the solution and open it.
2. In the Implementation Manager, click the Assemble and Download button and then Download a Copy.
3. Enter a description for the copy and save the .zip file locally.
   The status of the original remains In Development.
4. Optional: If you want to upload the copy to a different tenant, log on to this tenant.
5. In the Implementation Manager, click the Upload button and select the .zip file of the copy.
6. Agree to the terms of the upload disclaimer and click Continue.
   The copy is uploaded and opened. The status of the solution is In Development and you can continue developing.

Import Items of a Solution Template

After you have downloaded a solution template, you can import this template to add its items to any of your solutions.

For more information, see Solution Templates Quick Guide [page 149].

Troubleshooting

If you were not able to activate the solution or patch, try the following:

1. In the Implementation Manager, click Show Log File to display updated log information on the Version History tab.
   If your solution or patch did not contain any errors, click the Activate button to activate the solution or patch again.
2. If this does not work, click the Clean and Reactivate button in the Implementation Manager to clean all the objects in the solution or patch and generate them again.
3. If cleaning and reactivation fails, create an incident. For more information, see Report an Incident [page 262].
   Note that you can use the Reset to Last Active Version button in the Implementation Manager to retrieve the last active version of your solution or patch.

Further Tasks

Users with the SDK Administrator role can use the Administration toolbar to access a number of administrative tasks, such as creating and deleting customer-specific solutions or solution templates and managing user sessions.

For more information, see Administrator Quick Guide for Customer-Specific Solutions [page 144].

See Also

Create a Customer-Specific Solution [page 152]
4.7 Solutions and Solution Templates

4.7.1 Solution Templates Quick Guide for Customer-Specific Solutions

With a solution template you can organize common development content that you want to reuse for customer-specific solutions. Solution templates enable you to rapidly start the development of customer-specific solutions, for example, for a specific industry.

For more information, see Solution Templates [page 150].

Tasks

Create a Solution Template

You can create templates containing reusable items and upload these items to customer-specific solutions.

For more information, see here [page 151].

Create a Copy of a Solution Template

You can use a copy of your template, for example, to continue developing in order to try out different scenarios without disrupting the original. For this purpose, you download a copy of an existing template and then upload this copy to the same tenant or a different tenant.

1. In the SDK, log on to the tenant where you created the template and open it.
2. In the Implementation Manager, click the Assemble and Download button and then Download a Copy.
3. Enter a description for the copy and save the .zip file locally.
   The status of the original remains In Development.
4. Optional: If you want to upload the copy to a different tenant, log on to this tenant.
5. In the Implementation Manager, click the Upload button and select the .zip file of the copy.
6. Agree to the terms of the upload disclaimer and click Continue.
   The copy is uploaded and opened. The status of the template is In Development and you can continue developing.

Download a Solution Template

When you have finalized your template, you need to download the template before you can import its items to a solution.

1. In the SDK, log on to the tenant where you created the template, and open the template.
2. In the Solution Explorer, right-click the template and select Check In All Files.
3. On the Implementation Manager toolbar, click the Assemble and Download button.
4. Enter a file name to save the template as a .zip file locally.
The system checks the template for consistency and then downloads all the items in the template to the .zip file.

**Import Items of a Solution Template**

After you have downloaded a solution template, you can import this template to add its items to any of your solutions.

1. Log on to the tenant where you want to import the items of your template to a specific solution.
2. Create a new customer-specific solution or open an existing solution. For information about how to create a customer-specific solution, see [here](page 152).
   - The solution must have the status *In Development*.
3. In the **Implementation Manager**, click the **Import Solution Template** button and select the .zip file of the template that you have downloaded before.
   - The items of the template are added to the solution.
   - If the solution and the template contain items that have the same name, the system replaces the items in the solution by the items of the template.

Make sure that your solution works well with the replaced or added items, for example, check if the associations are still valid. When you have finalized your solution, activate it.

### 4.7.2 Solution Templates

With a solution template you can organize common development content that you want to reuse for customer-specific solutions. Solution templates enable you to rapidly start the development of customer-specific solutions, for example, for a specific industry.

- You can import a solution template in the SDK release version in which it has been created and in the subsequent release version.

The following restrictions and guidelines apply:

- You can create a solution template on a customer’s test tenant and on your development tenant.
- To create a solution template you need to be logged on with a user that has the **SDK Administrator** role.
- A template always has the status *In Development*. You cannot create a patch of a template.
- You are not able to define content using the key user tools/administrator tools, that is, you cannot enable the key user mode/administrator mode for a solution template. For information about the key user mode, see Solution Explorer [page 56] under **Solution**.
- You cannot create BC sets using SAP BCOs in a solution template.

You can create all items that you want to reuse in a solution, for example, business objects, actions, and events. When you have finalized your template, you can download it. You can then create a new solution or you can open an existing solution or patch and import the items of your template to this solution or patch. For more information, see Implementation Manager Quick Guide [page 146] under Import Items of a Solution Template.

- If you import a template into a solution, all items of the template are added to the solution. If items in the solution have the same name as in the template, the system replaces the solution’s items with the template’s items.
4.7.3 Create a Template for Customer-Specific Solutions

Overview

You can create templates containing reusable items and import these items into customer-specific solutions. This allows you, for example, to rapidly start the development of customer-specific solutions by reusing common development content, for example, for a specific industry.

The following restrictions and guidelines apply:

- You cannot create a template on a preproduction tenant.
- A template always has the status **In Development**. You cannot create a patch of a template.
- You are not able to define content using the key user tools/administrator tools, that is, you cannot enable the key user mode/administrator mode for a template.
  
  For information about the key user mode, see [Solution Explorer](#) under Solution.

- You cannot create BC sets using SAP BCOs in a template.

You can use a template in the SDK release version in which it has been created and in the subsequent release version.

Prerequisites

You are logged on to the repository of your development tenant or to the repository of your customer’s test tenant with a user that has the **SDK Administrator** role. For more information, see [Log On to the Repository](#) [page 261] and [User Roles](#) [page 38].

Procedure

1. On the **Administration** toolbar, click **Create Solution**. The **Create Solution** dialog box opens.
2. Enter the description and select the **Solution Template** type.
3. Select the deployment unit in which you want to create your template and click **OK**.
   
   For more information about deployment units, see [here](#) [page 163].

   Your new template is created and displayed in the **Solution Explorer**. The template name is created automatically and cannot be changed.

Result

You can create all items that you want to reuse in a solution, for example, business objects, actions, and events. When you have finalized your template, you can download it. You can then create a new solution or you can open an existing solution or patch and import the items of your template to this solution or patch. For more information, see [Solution Template Quick Guide](#) [page 149].

If you import a template into a solution, all items of the template are added to the solution. If items in the solution have the same name as in the template, the system replaces the solution’s items with the template’s items.
4.7.4 Create a Customer-Specific Solution

Overview

You can design and develop customer-specific solutions to enhance the standard SAP solution for a specific customer. In the SDK, you can create a solution in which all items that you develop or changes that you make to existing entities are stored.

Depending on the tenant that you are working on, the following applies:

- **Customer's Tenant**
  
  You create your solution for this specific customer on a test tenant or a preproduction tenant.

- **Development Tenant**
  
  On your development tenant you can develop solutions for different customers. Each time you create a solution, ensure that this solution is assigned to the correct customer ID. For more information, see Switch Customer [page 143].

Prerequisites

You are logged on to the repository of your development tenant or to the repository of a customer’s tenant with a user that has the SDK Administrator role. For more information, see Log On to the Repository [page 261] and User Roles [page 38].

Procedure

1. Optional: If you are developing on your development tenant, in the Administration menu choose Switch Customer.

   Check that the ID of the customer for which you want to develop the solution is displayed. If not, select a different customer or create a new customer assignment.

   ! If you are developing on your own development tenant, the solutions are always created for the customer that is currently displayed in the Switch Customer dialog box. Only a solution with the correct customer ID can then be uploaded to the corresponding tenant of your customer.

2. On the Administration toolbar, click Create Solution.

   The Create Solution dialog box opens.

   ! You can also use this function to create a solution template. For more information, see Create a Solution Template [page 151].

3. Enter the description and select the Customer-Specific Solution type.

4. Select the deployment unit in which you want to create your solution. For more information, see Deployment Unit [page 163].

   Your new solution is created and displayed in the Solution Explorer. It contains a project in which you create all items needed for your solution, for example, business objects, actions, and events.

   The solution name and project name are created automatically and cannot be changed. Your solution has the status In Development.
Exceptional Cases for Scalable Solutions

You should always create customer-specific solutions. However, the creation of scalable solutions is possible in exceptional cases. If you think you require a scalable solution, proceed as follows:

1. Create an incident and ask SAP Support to put it on component AP-RC-BDS-LM.
2. Enter the following information:
   - Solution name — use alphanumeric characters and underscores in the name
   - Solution description
   - Solution type (mashup, integrated solution, lightweight solution, add-on, or micro-vertical solution)
     For information about the types available, see https://wiki.sme.sap.com/wiki/x/aIKKEw.
   - Deployment unit of the solution — you can view the available deployment units in the Repository Explorer
   - Detailed explanation as to why you require a scalable solution

If SAP agrees to your request, you will receive instructions on how to proceed in the studio.

See Also

Mobile Solutions [page 151]
Lifecycle Management of Customer-Specific Solutions [page 127]

4.8 Patches for Customer-Specific Solutions

Overview

A patch is needed if you want to make changes or corrections to a customer-specific solution after it has been assembled. The customer-specific solution is then in maintenance mode and you can only make restricted changes for certain content types. For more information, see Maintenance Mode [page 154].

Only users with the SDK Administrator role can create patches.

Patch Types

Depending on the type of customer’s tenant where you are developing the creation and handling of a patch works differently:

- Creating a Patch on a Customer’s Test Tenant
  When you create a patch of a solution on a customer’s test tenant, the system creates a copy of your solution in a new namespace allowing you to make and test your changes without disrupting the original solution. This patch type is called a patch solution. When you want to test your patch solution in the test tenant, you first need to use the Implementation Manager in the SDK to enable the patch solution instead of the original solution in the test tenant.
  When you assemble the patch solution, the patch is copied back to the namespace of the original solution. The patch solution is used for all future updates to the original solution. Any changes you need to make in the future can be made by creating a new patch again. To do this, open the existing patch solution in the
Implementation Manager toolbar and click Create Patch. All future patches are created in the same patch solution but each patch has a different version number. For more information about which steps to perform to create a patch solution, see Create a Patch on a Customer’s Test Tenant [page 137].

- Creating a Patch on Your Development Tenant or on a Customer’s Preproduction Tenant
  The patch creation procedure on a preproduction tenant and your development tenant is basically the same. When you create a patch on one of these tenants, the system creates this patch in the same solution and namespace but with a different version number. For more information about which steps to perform, see Create a Patch on Your Development Tenant [page 141] or Create a Patch on a Customer’s Preproduction Tenant [page 134].

Distribution of Patches in the Production System

After you have activated a patch, the solution including the patch is available for productive use by the customer. If the solution is already running on other tenants in the same system as the production tenant, the patch is uploaded and activated on these tenants automatically. This happens in the background after the patch is successfully activated in the production tenant and may take some time. If the solution is running on other tenants in different systems, you need to upload the patch to these tenants manually.

To find out for which tenants the patch is intended, in the Implementation Manager, select the Tenant Status tab.

4.9 Maintenance Mode

Overview

A solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is Assembled. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer’s production tenant.

See Also

- Maintenance of Business Configuration Content [page 280]
- Maintenance of Analytics Content [page 335]
- Maintenance of Business Objects [page 305]
- Maintenance of Business Object Extensions [page 344]
- Patches for Customer-Specific Solutions [page 153]

4.10 Upgrade Information for Customer-Specific Solutions

The following information is relevant, if your customer’s system is upgraded to a new Feature Pack. The upgrade is maintained by SAP to ensure high quality upgrade of the customer’s production tenant. The upgrade does not have any impact on your customer-specific solution. However, be aware of the downtime period during the upgrade and follow the rules listed below.
During Upgrade
You cannot continue development during the upgrade process.

After the Upgrade
After the system has been upgraded, you can continue working on your solution. Be aware of the following:

- If the solution or the solution template still has the status *In Development* and you have activated its items before the upgrade, you must first activate the solution items after the upgrade again. Only then can you download the solution or the solution template.

- If you have downloaded a solution before the upgrade and then want to upload this solution to the upgraded system, you must first download the solution again. This makes your solution compatible with the new Feature Pack of the SAP solution.
5 Scripting Languages

5.1 Overview of Scripting Languages

Scripting Languages encompasses the following reference material:

<table>
<thead>
<tr>
<th>Get Informed About Scripting Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check out the reference material for:</td>
</tr>
<tr>
<td>• Scripting Language [page 156]</td>
</tr>
<tr>
<td>The scripting language is used to define the following:</td>
</tr>
<tr>
<td>○ Business objects [page 158]</td>
</tr>
<tr>
<td>○ Business object extensions [page 170]</td>
</tr>
<tr>
<td>It is also used to implement the business logic [page 178] for partner solutions. The methods provided for the implementation of the business logic are supplemented by the following built-in functions and reuse libraries:</td>
</tr>
<tr>
<td>○ Built-In Functions [page 228]</td>
</tr>
<tr>
<td>○ Reuse Libraries [page 202]</td>
</tr>
<tr>
<td>○ Reuse Libraries for Business Areas [page 223]</td>
</tr>
<tr>
<td>The reuse libraries provide basic functions that are used quite often, for example, the retrieval of context data, such as the current date or time of the current identity UUID.</td>
</tr>
</tbody>
</table>

Front-End Script for the User Interface Designer [page 240]
The front-end script is the programming language used in the UI designer to define a set of rules and the logic of the behavior of UI controls. For example, you can write a front-end script to enable an edit button only when a field is selected on the user interface.

Application Programming Interface for SDK Custom Panes and Custom Controls [page 433]
Learn about the API that enables you to access Controller.DataContainer to read the data from the actual running client component and register to events in case of changes. It also enables you to visualize data on your UI in the manner you choose.

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

5.2 Scripting Language for the SDK

5.2.1 Scripting Language Reference

Overview

The scripting language is used to define business objects and business object extensions as well as to implement the business logic for SAP Business ByDesign partner solutions. This lightweight language is easy to learn and to use with its focused set of features.

The sandbox environment for the scripting prevents a partner who has created erroneous implementations from damaging data of other users by accident. This kind of sandbox environment is ensured by the script execution runtime. Security-relevant features, such as direct access to database tables, are strictly controlled or not made
available in the scripting language. Instead, all data access is made available through the business object infrastructure.

The keywords, methods and functions of the scripting language are described in separate sections according to the task you want to perform:

- **Syntax for Business Object Definitions** [page 158]
  You use these keywords and functions to define the structure and interface of business objects.

- **Syntax for Business Object Extension Definitions** [page 170]
  You use these keywords and functions to define the structure and interface of business object extensions.

- **Syntax for Implementation of Business Logic** [page 178]
  You use these methods and keywords to implement the business logic of your solution.
  For this task, you can also use predefined built-in functions and reuse services. For more information, see the following documentation:
  - **Built-In Functions** [page 228]
  - **Basic Reuse Libraries** [page 202]
  - **Reuse Libraries for Business Areas** [page 223]

To fully understand the keyword descriptions, it is essential to have a basic understanding of the business object model and its features, such as associations, actions, and queries.

> Model names of SAP entities are in British English according to ISO 11179.

**Code Completion**

Based on the code you type, the code completion feature presents you with a list of options. For example, if you type a period (.), which separates the members in a path expression, the code completion is automatically activated and a drop-down list with the potential options is displayed. You can also use key combinations to display the valid code completion options, for example, `CTRL` + `SPACEBAR`. For more information, see the relevant syntax documentation under *Code Completion*.

**Syntax**

The syntax conventions for scripting language are as follows:

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>Indicates that you can use none, one, or more of the enclosed options. Do not include the brackets in your option.</td>
</tr>
<tr>
<td></td>
<td>A vertical line between two parts of a statement indicates that only one of these two parts can be used within the statement.</td>
</tr>
<tr>
<td>?</td>
<td>Indicates that the symbol (or the group of symbols in parenthesis) to the left of the operator is optional (it can appear just once or not at all). You can use it as a postfix to &quot;[ ]&quot;.</td>
</tr>
<tr>
<td>+</td>
<td>Indicates that something can be repeated at least once or any number of times. You can use it as a postfix to &quot;[ ]&quot;.</td>
</tr>
<tr>
<td>*</td>
<td>Indicates that something can appear any number of times (and possibly be skipped altogether). You can use it as a postfix to &quot;[ ]&quot;.</td>
</tr>
<tr>
<td>XXX</td>
<td>Indicates a placeholder. Replace placeholders with actual values in your code snippet.</td>
</tr>
</tbody>
</table>
5.2.2 Business Object Definitions

5.2.2.1 Syntax for Business Object Definitions

Use the keywords and functions listed below to describe the structure and interface of a business object in the SDK. For more information about the general syntax conventions and rules, see Scripting Language [page 156].

Code Completion

You can use `CTRL` + `SPACEBAR` at any cursor location to display the valid code completion options for the current context. If there is only one option, the code is completed at once. The code completion is also activated, if you enter a period (.) in a path expression or a colon (:), for example, in an `element` statement.

Naming Rules in Business Object Definitions

Comply with the following naming rules:

- The name of an entity must start with a letter and can contain letters, digits, and underscore characters. Do not start a name with “SAP_”.
- Related entities within a business object, for example, elements, associations, and actions, cannot have the same name as the business object. This does not apply for messages.
- Do not use the following technical terms as an entity name:
  - action
  - association
  - businessobject
  - element
  - import
  - namespace
  - node
  - query
  - to
  - type
  - using
  - valuation

Comments

You can enter comments using the following syntax:

- `comment line`
  - `// <comment text>`
  A comment line starts with two slashes (//) and spans to the end of the current line.
A comment block starts with a forward slash followed by an asterisk and ends with an asterisk followed by a forward slash.

List of Functions and Keywords

In the following, the functions and keywords are listed in alphabetical order:

- Action  [page 159]
- Annotation  [page 160]
- Association  [page 160]
- Business Object  [page 161]
- Dependent Object  [page 163]
- Deployment Unit  [page 163]
- Element  [page 164]
- Import  [page 165]
- Message  [page 166]
- Node  [page 166]
- Raises  [page 167]
- Valuation  [page 169]

List of Annotations

In the following, the annotations are listed in alphabetical order:

- Alternative Key  [page 160]
- Extension
- Label  [page 165]
- Multiple Texts  [page 166]
- Relevant For Access Control  [page 168]
- Tooltip  [page 168]
- Transient  [page 168]

5.2.2.2 Action (Business Object)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>action &lt;name&gt; [Raises]? ;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Use the <strong>action</strong> keyword to define an action for a node. Actions carry out the business logic and are implemented in .absl files.</td>
</tr>
</tbody>
</table>
5.2.2.3 Alternative Key (Business Object)

Syntax

| AlternativeKey | element <element name> : <data type>;

Description

If you want to identify a node uniquely with an identifier that can be read by human beings, you use the AlternativeKey annotation. You can use this identifier in an association to refer to a business object node. In addition, this identifier is stored as an additional key field in the database and thus accelerates enterprise searches and queries. It is not possible to have more than one alternative key of the same type in a business object node.

Custom business objects contain an implicit alternative key of type UUID for each node, enabling the node to be used as an association target.

Example

| AlternativeKey | element EmployeeID : ID;

See Also

Element (Business Object) [page 164]
Association (Business Object) [page 160]

5.2.2.4 Annotation (Business Object)

Syntax

| <annotation>[(<parameter>)??

Description

Annotations are used to add a category or property to an object specified by the subsequent keyword. The names of the annotations are predefined and can be displayed by the code completion. Technically, you can combine as many annotations as you want. But not all combinations are possible and not all of them make sense. For more information, see the list of possible combinations below.

Examples

| AlternativeKey | element EmployeeID : ID;
Example with annotation and parameter

| DeploymentUnit(CustomerRelationshipManagement) | businessobject BonusRule

{ ... ];
Example with two annotations

| DependentObject(TextCollection)] | [MultipleTexts] node TextCollection;

5.2.2.5 Association (Business Object)

Syntax

| <annotation> | association <name> [ <multiplicity> ?? to <composition path to target node> [using <alternative key>]??;
An association is a relationship between two business object nodes. The definition of an association includes a name, the multiplicity, a target node and, optionally, a target element. The target node can be either part of the same business object or part of a different business object. You can also refer to a target node or element in a different namespace. For more information, see Import [page 165].

Associations are based on a foreign key relationship. The using clause specifies the key element of the target business object node for the relationship. As a prerequisite, the key element must be an alternative key and foreign key usage must be allowed for it. The using clause is optional. For business objects defined in the same project, the target node’s hidden SAP_UUID element is used. For other business objects, the UUID of the target node is used.

The multiplicity of an association can be either [0..1] or [1..1]. If no multiplicity is specified, [1..1] is used by default. Only compositions can have a multiplicity with an upper boundary of “n”. However, you can easily simulate the [1..n] multiplicity by combining a [1..n] node composition with a [1..1] intra or cross-business-object association. For more information about compositions, see Node [page 166].

As a rule, the target node of an association must be located in the same deployment unit as the current business object or in the Foundation deployment unit.

The path to the target node consists of one or more compositions of the target business object. The first composition starts from its root node, the next composition starts from the target node of the first composition, and so on, until the association target node is reached.

If the target node is the root node, do not use the target node path in the association definition. If the target business object is the current business object, do not use the target business object in the association definition. However, this does not apply if the source and the target node is the root node of the current business object.

You can mark associations as relevant for access control by adding the annotation Relevant for Access Control [page 168].

### Examples

- **Association target within the current business object**
  
  Item is the name of the composition from the current node to the Item node.
  
  ```
  association ItemMainItem to Item;
  ```

  Item is the name of the composition from the current node to the Item node. SubItem is the name of the composition from the Item node to the SubItem node.

  ```
  association ItemToSubItem to Item.SubItem;
  ```

- **Association to different business object**

  ```
  association ToServReq to AP.CRM.Global:ServiceRequest.ServiceRequestOverview using ID;
  ```

- **Using keys and root node as a target node**

  The default key of target root node is used.

  ```
  association Rule[0..1] to BonusRule;
  ```

  The key of target root node is specified.

  ```
  association Material to Product using ProductID;
  ```

- **Target with fully qualified business object name**

  ```
  association Customer [0..1] to AP.FO.BusinessPartner.Global:Customer using ID;
  ```

- **Target with fully qualified business object name using an alias**

  ```
  import AP.FO.BusinessPartner.Global as bp;
  ```

  ```
  association Customer[0..1] to bp:Customer using ID;
  ```

- **Association marked as relevant for access control**

  ```
  import AP.FO.BusinessPartner.Global as bp;
  ```

  ```
  [RelevantForAccessControl] association ToBP to bp:BusinessPartner;
  ```

### See also

- Alternative Key (Business Object) [page 160]
- Valuation (Business Object) [page 169]

### 5.2.2.6 Business Object (Business Object)

#### Syntax

```
businessobject <name> [ <Raises> ]? { [〈Message〉? 〈Node〉] };
```
Use the `businessobject` keyword to specify the name and the nodes of a business object. A business object consists of a root node, which can contain child nodes. Even if not specified, the multiplicity of the root node is `[1..1]`. For more information, see Node [page 166].

The `.bo` file that contains the business object definition must have the same name as the business object itself. Each file contains the definition of a single business object.

<table>
<thead>
<tr>
<th>Example</th>
<th>Business object, message and root node element. The root node raises a specified message:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>businessobject BonusRule raises ReferenceInvalid { message ReferenceInvalid text &quot;Reference is invalid&quot;; element ItemID : ID; }</code></td>
</tr>
</tbody>
</table>

See also Message [page 166]

5.2.2.7 Cross Deployment Unit (Business Object)

**Syntax**

```
[CrossDeploymentUnit] association <name> to <path to target node> [using <composition path to target node]?]
```

**Description**

You can use the `CrossDeploymentUnit` annotation to define an association from a node of a business object in one deployment unit to a node of a business object in another deployment unit. Optionally, you can specify the `AlternativeKey` of the target node with the `using` keyword.

**Examples**

```
[CrossDeploymentUnit] association ToBusinessPartner to BusinessPartner;
[CrossDeploymentUnit] association ToSalesOrder to SalesOrder using ID;
```

See also Association [page 160]

5.2.2.8 Default Values (Business Object)

**Syntax**

Basic data types:

```
element <element name> : <data type name> = <default value>;
```

Structured data types:

```
element <element name> : <data type name> = {<component1> = <default value>, ..., <componentX> = <default value>};
```

You cannot use aggregated data types.

**Description**

Default values are initial values that are used to fill fields automatically when a new instance is created. The SDK supports only static default values such as Boolean values, status information, numeric values, and free text. There is no syntax check for default values included in quotation marks.

Translation of default values is not supported.

**Examples**

Basic data types:

```
element Text1 : {content = "14", languageCode = "EN"};
element Text2 : LANGUAGEINDEPENDENT MEDIUM_Description = "Test Text";
element Date : Date = "2012–11–01";
element Time : Time = "23:59:59";
element Integer : IntegerValue = 12345;
element Code :ActionCode = "01";
```

Structured data type:

```
element Amount : Amount = {content = 13.45, currencyCode = "EUR"};
```
5.2.2.9 Dependent Object (Business Object)

**Syntax**

```
[DependentObject(<dependent object name>))] [MultipleTexts]? node <name>;
```

**Description**

A dependent object is a reusable part of a business object that, from a business point of view, cannot stand alone, but can only be used in the context of the business object. You can include the `TextCollection` and `AttachmentFolder` dependent objects in a business object definition.

- For the `TextCollection` dependent object you can use the `MultipleTexts` annotation to support different text types.
- In order to define the dependent object inclusion node, you have to specify a name. You can either use the dependent object name itself, for example, `TextCollection`, or add a specific meaning to the dependent object name, for example, `ProcurementSpecificationTextCollection`. If you include a dependent object in a business object definition several times, you should provide different names for each of the dependent object inclusion nodes. You can, for example, use the following syntax: `<subordinate node name>` + `<dependent object name>`, such as `ItemAttachmentFolder`.
- You can add the same dependent object only once to a business object node.

**Example**

```
[DependentObject(TextCollection)] node ItemTextCollection;
```

**Example with two annotations**

```
[DependentObject(TextCollection)] [MultipleTexts] node TextCollection;
```

See also: Annotation [page 160]

5.2.2.10 Deployment Unit (Business Object)

**Syntax**

```
[DeploymentUnit(<deployment unit name>))] [businessobject <name>] [ <Raises> ]? 
{ [ <message>? <node> ] }; 
```

Cross-deployment-unit association:

```
[CrossDeploymentUnit] [association <name>] to <composition path to target node> 
[using <alternative key>]?;
```
**Description**

A deployment unit is a piece of software that can be operated on a separate physical system, isolated from other pieces of software. The SAP solution is organized in deployment units that group semantically related business objects, for example, Customer Relationship Management. There is also a special Foundation deployment unit that mainly contains master data objects, for example, Material, Employee, and Customer.

Each business object resides in a deployment unit. You can explore SAP business objects that are released with the public solution model (PSM) and the deployment units in the Repository Explorer tool window. For more information, see [Repository Explorer](page 53).

When a solution is created, the solution has to be assigned to a deployment unit. The business objects that are created subsequently in the solution are assigned to this default deployment unit. If you want to assign a business object to a different deployment unit, use the annotation `DeploymentUnit` at the beginning of the business object definition.

You cannot create your own deployment units.

You can directly access business objects that reside in the Foundation deployment unit from any deployment unit. However, business objects in the Foundation deployment unit cannot access any business object residing in another deployment unit, not even by using asynchronous communication. You can only use a query to read the data of this business object.

For all other deployment units the following applies: You cannot directly change the data of a business object residing in another deployment unit. However, you can read the data of such a business object by using a query, the `Retrieve` method or an association. To change the data of a business object residing in another deployment unit, use asynchronous, message-based communication. For more information, see [Configure Internal Communication](page 418).

You cannot use a cross-deployment-unit association to access a business object in another deployment unit. You can only use a cross-deployment-unit association in an analytical data source to collect data from a business object residing in a different deployment unit.

**Example**

Assigning a business object to a different deployment unit:

```java
[DeploymentUnit(CustomerRelationshipManagement)] businessobject BonusRule {
    elementID : ID;
}
```

**See also** [Association](page 160), [Business Object](page 161)

### 5.2.2.11 Element (Business Object)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[ &lt;annotations&gt; ]? element &lt;name&gt; : &lt;data type name&gt;;</code></td>
<td>Use the <code>element</code> keyword to define an element of a business object node. The element name must be unique within the node. The element type can be a simple data type, for example, ID, or it can be a structured data type such as Amount. You can import data types from different namespaces. However, you cannot define new data types. For more information on importing entities, see [Import](page 165). When defining an identifier element, we recommend you use the ID data type instead of the Identifier data type. The ID data type always has 60 characters, is not structured and does not contain the Content element. The data type supports, for example, upper case conversion and alpha conversion. Alpha conversion fills purely numeric user input like “4711” with leading zeros from the left side to allow better sorting in character fields. For example, if alpha conversion is not used, an alphabetic sorting would lead to results such as: “1”, “10”, “100”, “1000”, “2”, “20”, “200”, and so on. You can define additional properties of an element by using annotations. For example, the annotation <code>AlternativeKey</code> defines the key element of a business object node. Key elements uniquely identify a node and can be used to identify the target of an association. An alternative key, <code>SAP_UUID</code> of type UUID, is generated implicitly for each node. Therefore, you do not need to define an alternative key for each node. Enclose annotations in square brackets.</td>
</tr>
</tbody>
</table>

© 2012 SAP AG. All rights reserved. • PUBLIC
5.2.2.12 Import (Business Object)

**Syntax**

```
import <namespace name>;
```

**Description**

Use the import keyword to import business objects and data types from namespaces that are available in the repository. The namespace name is typically a Uniform Resource Identifier (URI) in dot notation (see example below).

You can use imported business objects and data types in the definition of associations, elements, and messages. You can reference the imported objects directly by their names instead of using fully qualified names. If you use a namespace only for one or two sub-entities, you can omit the import statement and use the qualified name `<namespace name>::<name>` instead. If there are no conflicts for any entity in the business object within all imported namespaces, you can omit the relevant namespace, as the system can identify it automatically.

If you use more than one namespace in your business object definition, we recommend that you define an alias for each imported namespace and qualify business objects or data types from these namespaces by using the aliases.

You do not need to import business objects and data types from the namespace of the current project but you can access them directly by their names.

Example Import of a namespace:

```
import AP.FO.BusinessPartner.Global;
```

Importing a namespace and defining an alias:

```
import AP.Common.GDT as apc;
import ESF as esfnsp;
```

Using the defined aliases:

```
element duration : apc:Duration;
message ReferenceInvalid text "Reference & is invalid": apc:NumberValue;
association Customer to esfnsp:BusinessPartner;
```

**See also** Element [page 164], Association [page 160]

5.2.2.13 Label (Business Object)

**Syntax**

```
[Label ("<label name>")]
```

**Description**

You can define a label for a business object field. If you then add the field to a screen, the label is displayed on the screen instead of the field name.

You can translate the label text by exporting it to an XLFF file. For more information, see the Translation Quick Guide [page 263].

**Example**

```
[Label ("My label for field 1")]
```

```
element ROOT_Field1 : Text;
```
5.2.2.14 Message (Business Object)

| Syntax | message <message ID> text "<message text>" [: <data type 1>, <data type 2>, <data type 3>, <data type 4>]]?]?]?]? ; |
| Description | Use the message keyword to define a message between two business objects. The definition of a message includes the message ID, the message text, and, optionally, data types of up to four message parameters. The parameter names are defined by the system. The limit for a message ID is 30 characters and for a message text is 73 characters. The message ID must be unique within the business object; use uppercase for letters. You can reference data types from different namespaces. For more information, see Element [page 164] and Import [page 165]. |
| Example | Message without parameters: message DateMissing text "Due date of invoice is missing"; Message with two parameters: message ReferenceInvalid text "Reference to sales order &1 invalid for &2" : IntegerValue, ESF:AddressType; |
| See also | Business Object [page 161], Node [page 166], Raises [page 167] |

5.2.2.15 Multiple Texts (Business Object)

| Syntax | [DependentObject(<dependent object name>)) [MultipleTexts]? node <name>; |
| Description | You can use the MultipleTexts annotation together with the TextCollection dependent object to support different text types. If the TextCollection dependent object is assigned to the current node, the MultipleTexts annotation defines if more than one text is allowed. |
| Example | [DependentObject(TextCollection)] [MultipleTexts] node TextCollection; |

See Also
Node [page 166]
Dependent Object [page 163]

5.2.2.16 Node (Business Object)

| Syntax | node <name> [ <multiplicity> ]? [ <Raises> ]? {[ <Element>, <Action>, <Association>, <Node> ]* } |
Use the `node` keyword to define a business object node and its features, that is, elements, actions, node associations, child nodes, and raised messages. The name of the node must be unique within the business object. For the business object, the root node of the node hierarchy is automatically created. The multiplicity of the root node is `[1..1]`. Each direct or indirect child node implicitly contains a `ToRoot` association. A node must contain at least one explicitly defined element (not including `SAP_UUID`, which is implicitly generated for each node). Elements, actions, associations, and child nodes must have unique names within the node.

The definition of a child node implicitly introduces a composition, that is, a relationship between a parent node and a child node. The multiplicity of the child node defines the upper and lower bound of that composition, that is, how many instances of the child node can be part of the parent node. The multiplicity of a node can be `[0..1]`, `[1..1]`, `[0..n]`, or `[1..n]`. If no multiplicity is defined, the multiplicity `[0..1]` is used by default.

You can use the composition to access features of the child node from the parent node (for example, in action implementations). For each composition, the reverse `ToParent` association is created implicitly. `SelectAll` and `ByElements` queries are created automatically for each node. It is not possible to define new queries in the business object.

**Example**

Business object and `raises` statement for the root node, message definition and root node element as well as subnode with a `raises` statement followed by a subnode element:

```
businessobject WorkingPlan raises DateMissing {
  message DateMissing text "Date is missing";
  element ItemID : ID;
  node Item [0..n] raises BonusRule.InvalidReference {
    element ItemID : ID;
  }
}
```

See also [Element](page 164), [Association](page 160), [Action](page 159), [Message](page 166), [Raises](page 167)

### 5.2.2.17 Raises (Business Object)

**Syntax**

```
raises <message ID> [, <message ID>][,...]? ;
```

**Description**

At least one message ID must follow the `raises` statement. Following the message ID, you can enter as many messages as needed. If the message is not part of the same business object, you must also specify the message group. The name of the message group is identical to the corresponding name of the business object.

**Example**

- `Raises` statement that raises a message from the same business object:

  ```
businessobject BonusRuleOne raises DateMissing {
    message DateMissing text "Date is missing";
    element ItemID : ID;
  }
  ```

- `Raises` statement that raises messages from the same business object and messages with different message groups:

  ```
businessobject BonusRuleTwo raises BonusRuleOne.DateMissing {
    message ReferenceInvalid text "Reference is invalid";
    element ItemID : ID;
    node Item raises BonusRuleOne.DateMissing, ReferenceInvalid {
      element ItemID : ID;
    }
    action Count raises ReferenceInvalid;
  }
  ```

See also [Message](page 166), [Node](page 166) (see examples)
5.2.2.18 Relevant for Access Control (Business Object)

<table>
<thead>
<tr>
<th>Syntax</th>
<th><code>[RelevantForAccessControl[(&lt;parameter&gt;))]?]</code></th>
</tr>
</thead>
</table>
| Description | You can implement instance-based access control for a business object you created by referencing an SAP business object for which an access context is defined. To do this, you mark associations as relevant for access control by adding the annotation `RelevantForAccessControl`. The following conditions must be met:  
  - For each business object, you can only use one association that is relevant for access control.  
  - The source node of the association must be the root node.  
  - The target node of the association must be root node of the target business object.  
  - The target business object must have an association that is relevant for access control. The target of this association must be the dependent object `Access Control List`.  |
| Example | `[RelevantForAccessControl] association toBusinessPartner to BusinessPartner;` |
| See also | Define Access Control [page 309] |

5.2.2.19 Tooltip (Business Object)

<table>
<thead>
<tr>
<th>Syntax</th>
<th><code>[Tooltip (&lt;tooltip name&gt;))] element &lt;name&gt; : &lt;data type&gt;;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>You can define a tooltip for a business object field. If you then add the field to a screen, the label is displayed on the screen instead of the field name. You can translate the tooltip text by exporting it to an XLFF file. For more information, see the Translation Quick Guide [page 263].</td>
</tr>
<tr>
<td>Example</td>
<td><code>[Tooltip (&quot;My tooltip for field 1&quot;)] element ROOT_Field1 : Text;</code></td>
</tr>
</tbody>
</table>

See Also

Add a Label and Tooltip to a Business Object [page 316]
Annotation [page 160]

5.2.2.20 Transient (Business Object)

<table>
<thead>
<tr>
<th>Syntax</th>
<th><code>[Transient] element &lt;name&gt; : &lt;data type&gt;;;</code></th>
</tr>
</thead>
</table>
| Description | You can use the `Transient` annotation to create transient fields. The data in such a field is not persistent, this means, it is not saved in the database. A transient field is filled after an `AfterLoading` event has been executed. The Transient annotation can be combined with the following annotations:  
  - Label [page 165]  
  - Tooltip [page 168]  |
| Example | `[Transient] element myTransientField : Text;`  
  `[Transient][Label("My transient field")] element myTransientField : Text;` |
5.2.2.21 Valuation (Business Object)

Syntax

```
association <name> [<multiplicity>]
    to <composition path to target node>
    valuation (<element of target node> <operator> <element defined in source node>);
```

Description

The `valuation` keyword enables you to create 0..n associations and 1..n associations. Thus, you can model hierarchical business object structures, for example, a product catalog. You can add the `valuation` keyword to an association in order to evaluate the elements of the target node. A `valuation` association must have the multiplicity [0..n] or [1..n]. Usually, associations represent relations between nodes of the same business object or between nodes of different business objects. However, when you use the `valuation` keyword, the association must refer to a node within the same business object. Depending on the node type, the valuation is executed as follows:

- If the association refers to a `sub-node` of a business object, the valuation is executed on the sub-node instances that belong to the same business object instance as the source node instance (association within a business object instance).
- If the association refers to the `root node` of a business object, the valuation is executed on all instances of this business object (association across all instances of a business object).

The `valuation` keyword is followed by a condition, which is evaluated when the association is resolved. You can define the condition as follows:

1. You specify the target element. This element must exist in the target node of the association.
2. You specify the comparison operator `==` (is equal to).
3. You specify an element that is defined in the source node.

In a condition, you can specify any non-structured data type.

You cannot use the `valuation` keyword together with the `using` keyword and the `RelevantForAccessControl` annotation.

Examples

**Valuation within a business object instance:**

```
businessobject BusinessObject1 {
    [AlternativeKey] element ID:ID;
    node ChildHierNode[0..n] {
        element ID:ID;
        element ParentID:ID;
        association toChild [0..n] to ChildHierNode valuation (ParentID == ID);
    }
}
```

In the real world, `BusinessObject1` may represent a catalog, containing sub-items that are ordered in a hierarchy (all items belong to the same catalog)

**Valuation across all instances of a business object:**

```
businessobject BusinessObject1 {
    [AlternativeKey] element ID:ID;
    element ParentID:ID;
    association toChild [0..n] to BusinessObject1 valuation (ParentID == ID);
}
```

In the real world, `BusinessObject1` may represent persons or objects (for example, buildings) that build a hierarchy.

See also

`Association` [page 160]
5.2.3 Business Object Extension Definitions

5.2.3.1 Syntax for Business Object Extension Definitions

Use the keywords and functions listed below to describe the structure and interface of a business object extension in the SDK.
For more information about the general syntax conventions and rules, see Scripting Language [page 156].

Naming Rules in Business Object Extensions

Comply with the following naming rules:

- The name of an entity must start with a letter and can contain letters, digits, and underscore characters only. Do not start a name with “SAP”.
- Entities within a business object cannot have the same name as the business object.
- Entities cannot use names that already exist in the business object being extended.
- Do not use the following technical terms as an entity name:
  - action
  - association
  - businessobject
  - element
  - import
  - namespace
  - node
  - query
  - to
  - type
  - using

Extension Template

When you create a business object extension, you are presented with a template that includes the extendable nodes. Taking the Sales Order business object as an example, the extension template begins with the annotation [Extension] and contains the extendable node Item. The Root node is also extendable, although not explicitly shown.

```plaintext
import AP.Common.GDT;
import AP.CRM.Global;

[Extension] businessobject AP.CRM.Global:SalesOrder { // You must activate this business object before you can access the extension fields // in script files, forms, and screens
node Item {
```
Code Completion

Based on the code you type, code completion presents you with a list of valid options. For example, when you type a colon (:) after an element name, you will see a list of valid data types. If there is only one valid option, the code is completed at once.

You can also use CTRL + SPACEBAR to force the code completion function to display the valid options at the current cursor location.

Comments

You can enter comments using the following syntax:

- **comment line**
  
  ```
  // <comment text>
  ```

  A comment line starts with two slashes (//) and spans to the end of the current line.

- **comment block**
  
  ```
  /* <comment text>
  <comment text>
  <comment text>
  */
  ```

  A comment block starts with a forward slash followed by an asterisk and ends with an asterisk followed by a forward slash.

List of Keywords

- Business Object [page 172]
- Element [page 173]
- Import [page 174]
- Node [page 175]
- Message [page 174]
- Raises [page 175]

List of Annotations

- Decimal [page 172]
- Label [page 174]
- Reference [page 176]
- Relation [page 177]
- Scenario [page 177]
- Tooltip [page 178]
5.2.3.2 Business Object (Business Object Extension)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>[Extension] businessobject &lt;name&gt; { &lt;Node&gt;* };</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The businessobject keyword specifies the business object being extended.</td>
</tr>
</tbody>
</table>

| Example | [Extension] businessobject AP.CRM.Global:SalesOrder { node Item { } } |

5.2.3.3 Decimal (Business Object Extension)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>[Decimal (x,y)] element &lt;name&gt; : DecimalValue;</th>
</tr>
</thead>
</table>
| Description | You can declare a decimal value with a specific number of digits before and after the decimal point. The decimal annotation [Decimal (x,y)] requires two parameters:  
- x = the total number of digits allowed, up to a maximum of 29  
- y = the number of digits allowed after the decimal point, up to a maximum of 14

The first parameter value must be greater than the second parameter value.

When you use negative numbers, the minus sign is counted as a digit. For example, [Decimal (2,0)] allows 55 and -5 but not -55. |

<table>
<thead>
<tr>
<th>Example</th>
<th>Valid Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Decimal (2,1)] element Test_Decimal_01 : DecimalValue;</td>
<td></td>
</tr>
<tr>
<td>- possible numbers: 5, 5.5, -5.5, 0.5</td>
<td></td>
</tr>
<tr>
<td>[Decimal (6,2)] element Test_Decimal_02 : DecimalValue;</td>
<td></td>
</tr>
<tr>
<td>- possible numbers: 55555.55, -555.55, 555555, 0.55</td>
<td></td>
</tr>
<tr>
<td>Invalid Syntax</td>
<td></td>
</tr>
<tr>
<td>[Decimal (4,4)] element Test_Decimal_03 : DecimalValue;</td>
<td></td>
</tr>
<tr>
<td>[Decimal (6,8)] element Test_Decimal_04 : DecimalValue;</td>
<td></td>
</tr>
</tbody>
</table>

5.2.3.4 Default Values (Business Object Extension)

| Syntax | Basic data types: element <element name> : <data type name> = <default value>;  
Structured data types: element <element name> : <data type name> = {<component1> = <default value>, <componentX> = <default value>};  
You cannot use aggregated data types. |

Default values are initial values that are used to fill fields automatically when a new instance is created. The SDK supports only static default values such as Boolean values, status information, numeric values, and free text. There is no syntax check for default values included in quotation marks.

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic data types:</td>
</tr>
<tr>
<td>element Text1 : {content = &quot;14&quot;, languageCode = &quot;EN&quot;};</td>
</tr>
<tr>
<td>element Text2 : LANGUAGEINDEPENDENT_MEDIUM_Description = &quot;Test Text&quot;;</td>
</tr>
<tr>
<td>element Date : Date = &quot;2012-11-01&quot;;</td>
</tr>
<tr>
<td>element Time : Time = &quot;23:59:59&quot;;</td>
</tr>
<tr>
<td>element Integer : IntegerValue = 12345;</td>
</tr>
<tr>
<td>element Code : ActionCode = &quot;01&quot;;</td>
</tr>
<tr>
<td>[Decimal(12.5)] element MyOKDecimal2 : DecimalValue = 1234.12345</td>
</tr>
<tr>
<td>Structured data type:</td>
</tr>
<tr>
<td>element myOKAmount1 : Amount = {content = 13.45, currencyCode = &quot;EUR&quot;};</td>
</tr>
</tbody>
</table>

Translation of default values is not supported.

See also
- Element [page 173]
- Data Types [page 78]

5.2.3.5 Element (Business Object Extension)

Syntax

```plaintext
[ <Annotation> ]? element <name> : <data type>;
```

Use the `element` keyword to define an extension element for a business object node. The element name must be unique within the solution and cannot exceed 60 characters. The element type can be a simple data type, for example, `Text`, or it can be a structured data type such as `Amount`. You can import data types from different namespaces. However, you cannot define new data types. For more information on importing entities, see `import`.

**Data Types**

The following data types are available by default:
- `Amount`
- `Date`
- `DecimalValue`
- `CodeList – SAP code lists`
- `ID`
- `Identifier – SAP identifiers`
- `Indicator`
- `Quantity`
- `Text`
- `Time`
- `WebURI`
- `EmailURI`

By importing additional namespaces, you can access additional SAP CodeLists and Identifiers. You can also use CodeLists that you have defined within your solution.

**Example**

```plaintext
element Target : Amount;
```
5.2.3.6 Import (Business Object Extension)

**Syntax**

```plaintext
import <namespace name>;
```

**Description**

Use the `import` keyword to import data types from namespaces that are available in the repository. The namespace name is typically a Uniform Resource Identifier (URI) in dot notation (see example below). You can use imported data types in the definition of elements. You can reference the imported data types directly by their names instead of using fully qualified names. If you use a namespace only for one or two sub-entities, you can omit the `import` statement and use the qualified name `<namespace name>:<name>` instead. If there are no conflicts for any entity in the business object within all imported namespaces, you can omit the relevant namespace, as the system can identify it automatically.

You can also define an alias for an imported namespace and qualify data types from this namespace by using the alias.

**Example**

Import of a namespace:

```plaintext
import AP.Common.GDT;
```

Definition of an alias:

```plaintext
import AP.Common.GDT as APC;
```

Using the defined alias:

```plaintext
element Target : APC:Amount;
```

5.2.3.7 Label (Business Object Extension)

**Syntax**

```plaintext
[Label ("<label name>")] element <name> : <data type>;
```

**Description**

You can define a label for an extension field. If you then add the extension field to a screen, the label is displayed on the screen instead of the extension field name.

**Example**

```plaintext
[Label ("Screen Label")] element SalesOrder_BO_extension_field : Text;
```

5.2.3.8 Message (Business Object Extension)

**Syntax**

```plaintext
message <message ID> text "<message text>" [: <data type 1>[, <data type 2>[, <data type 3>][, <data type 4>]]?]?]??;
```

**Description**

Use the `message` keyword to define a message for a business object extension. The definition of a message includes the message ID, the message text which can include up to four parameters, and the data types of any parameters used.

The limit for a message ID is 30 characters and for a message text is 73 characters. The message ID must be unique within the business object extension. Messages can only be raised in the business object extension where they are defined. In addition, messages must be assigned to the individual nodes where they are to be called.
Example

```java
import AP.Common.GDT;
import AP.CRM.Global;
[Extension] businessobject AP.CRM.Global:Opportunity raises
Information_Message, Error_Message, Warning_Message, Success_Message {
    // You must activate this business object before you can access the extension fields
    // in script files, forms, and screens.
    message Information_Message text "This is an Information message";
    message Error_Message text "This is an Error message with one parameter &1" : LANGUAGEINDEPENDENT_SHORT_Name;
    message Warning_Message text "This is a Warning message with two parameters &1 and &2" : LANGUAGEINDEPENDENT_SHORT_Name, IntegerValue;
    message Success_Message text "This is a Success message with a variable parameter &1" : Amount;
    element Test_Extension_Field : Amount;
    node Item raises Information_Message {
    }
}
```

### 5.2.3.9 Node (Business Object Extension)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>node &lt;name&gt; [ &lt;multiplicity&gt; ]? [{&lt;Element&gt;, &lt;Node&gt;}* ];</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The <code>node</code> keyword specifies the extendable nodes within a business object. You cannot create your own nodes within a business object extension.</td>
</tr>
</tbody>
</table>
| Example                 | [Extension] businessobject AP.CRM.Global:SalesOrder {
    node Item {
    }
} |

### 5.2.3.10 Raises (Business Object Extension)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>raises &lt;message ID&gt; [, &lt;message ID&gt;[,...]??];</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Use the <code>raises</code> keyword to declare messages for a business object extension. At least one message ID must follow the <code>raises</code> statement. Messages can only be raised in the business object extension where they are defined. In addition, messages must be assigned to the individual nodes where they are to be called.</td>
</tr>
</tbody>
</table>
Example

```java
import AP.Common.GDT;
import AP.CRM.Global;
[Extension] businessobject AP.CRM.Global:Opportunity raises
Information_Message, Error_Message, Warning_Message, Success_Message {
    // You must activate this business object before you can access the extension fields
    // in script files, forms, and screens.
    message Information_Message text "This is an Information message";
    message Error_Message text "This is an Error message with one parameter &1" : LANGUAGEINDEPENDENT_SHORT_Name;
    message Warning_Message text "This is a Warning message with two parameters &1 and &2" : LANGUAGEINDEPENDENT_SHORT_Name, IntegerValue;
    message Success_Message text "This is a Success message with a variable parameter &1" : Amount;
    element Test_Extension_Field : Amount;
    node Item raises Information_Message {
    }
}
```

5.2.3.11 Reference (Business Object Extension)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>[Reference] element &lt;extension_field_name&gt;;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The Reference annotation enables you to use an element declared in one business object extension in another business object extension. You can also use an element in the item node that is declared in the root node of the same business object extension. The Reference annotation cannot be combined with the following annotations:</td>
</tr>
<tr>
<td></td>
<td>● Decimal</td>
</tr>
<tr>
<td></td>
<td>● Relation</td>
</tr>
<tr>
<td></td>
<td>● Label</td>
</tr>
<tr>
<td></td>
<td>● Tooltip</td>
</tr>
<tr>
<td></td>
<td>● Transient</td>
</tr>
<tr>
<td></td>
<td>You do not need to include a data type when referencing an element, as the data type is specified in the original element definition.</td>
</tr>
</tbody>
</table>

Examples

1. **Reference from one business object extension to another business object extension**
   Definition of the first business object extension:
   ```java
element Test_Extension_Field : Text;
```
   Definition of the second business object extension:
   ```java
[Reference] element Test_Extension_Field;
```

2. **Reference from an item node to the root node of the business object extension**
   Definition of the root node:
   ```java
element Test_Extension_Field : Text;
```
   Definition of the item node:
   ```java
[Reference] element Test_Extension_Field;
```

In contrast to the Relation [page 177] annotation, which refers to any business object element, the Reference annotation refers to a specific element within the same business object.
5.2.3.12  Relation (Business Object Extension)

Syntax

<table>
<thead>
<tr>
<th>[Relation(Namespace:BusinessObject.BO_Node.Element)] element &lt;data type&gt;;</th>
</tr>
</thead>
<tbody>
<tr>
<td>with &lt;data type with representation term Identifier&gt;</td>
</tr>
</tbody>
</table>

Description

The `Relation` annotation enables you to create a reference to a business object by using the identifier of an element, for example, `ID`, `PartyID`, or `UUID` as the target of the reference. Thus, you can create an element that is like the element you are referring to and retrieve the data of this element, for example, for an object value selector or a query. Tooltips and labels are not retrieved from the element you are referring to. You need to specify the namespace in the annotation, if the namespace has not yet been specified in the definition of the business object extension.

You can combine the `Relation` annotation with the `Scenario`, `Label`, and `Tooltip` annotations and with default values but not with the `Reference`, `Decimal`, and `Transient` annotations.

Examples


5.2.3.13  Scenario (Business Object Extension)

Syntax

| [Scenario (<scenario_name>)] element <extension_field_name>:<data_type>; |

Description

A process extension scenario links the data from one business context to other related business contexts. You create an extension scenario from a list of predefined extension scenarios. For example, you can create an extension scenario that links the following business contexts:

- **Lead - General Information > Opportunity - General Information**
- **Opportunity - General Information > Sales - General Information**

Each extension scenario contains one or more data flows. Each data flow consists of a source and target business context. This reflects the direction in which data is passed from one business context to the next as part of a business process.

- **Opportunity - General Information > Sales - General Information**
  - Opportunity - General Information > Sales Quote - General Information
  - Opportunity - General Information > Sales Order - General Information

Extension scenarios are specific to the node for which they were created. If an extension scenario was created for the root node, then it can only be called at the root node within the `.xbo`. Similarly, an extension scenario created for the item node can only be called from the item node within the `.xbo`.

If you now extend any of the business objects involved in the scenario, you can use the annotation `[Scenario]` to declare that the element is available to all the business contexts specified in the scenario. For example, if you add an element to the Lead business object, you can use an extension scenario to make this element available if you extend the Opportunity, Sales Quote, or Sales Order. When an Opportunity is created from the Lead, the data for your extension field is passed to the Opportunity. Similarly, when a Sales Order is created from the Opportunity, the data is passed to the Sales Order.

Example

| [Scenario (Sales_Lead_Tracking)] element Sales_Lead_Source : Text; |

See Also

For a list of all predefined extension scenarios, see here [page 357].
5.2.3.14 Tooltip (Business Object Extension)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>[Tooltip (&lt;&quot;tooltip name&quot;&gt;)]) element &lt;name&gt; : &lt;data type&gt;;;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>You can define a tooltip for an extension field. If you then add the extension field to a screen, the tooltip is displayed when you place the cursor on the extension field.</td>
</tr>
<tr>
<td>Example</td>
<td>[Tooltip (&quot;Screen Tooltip&quot;)]) element SalesOrder_BO_extension_field : Text;</td>
</tr>
</tbody>
</table>

5.2.4 Business Logic

5.2.4.1 Syntax for Implementation of Business Logic

The methods and keywords listed below enable you to implement the business logic for your solution in script files, such as actions, validations and events. You can implement, for example, BeforeSave events and OnSave validations, and the actions that you have defined for your business objects. For more information about the general syntax conventions and rules, see Scripting Language [page 156].

You can also use built-in functions and reuse libraries that extend the scope of the methods described below. For more information, see the following documentation:

- Built-In Functions  [page 228]
- Basic Reuse Libraries  [page 202]
- Reuse Libraries for Business Areas  [page 223]

Type Handling

The scripting language supports type inference, thereby avoiding the need of explicitly identifying relevant types. This is especially useful for script files, where elements of existing business objects are accessed either by your own business objects or by SAP business objects. For structured element types, the code completion enables you to see the data type's structure while accessing the elements of the involved business objects.

For more information about the basic data types that the scripting language supports and their behavior, see Behavior of Data Types  [page 235].

Code Completion

Apart from using a period (.) for the code completion of path elements, you can use the key combination CTRL + J to display all options for the current context. You can use this feature at any cursor location, for example, when the cursor is located at the end of a partially typed identifier or keyword or when the cursor is located within or directly before an already existing and valid identifier.
Method Tooltip

The method tooltip displays the documentation of a method and its parameter definitions. The method tooltip appears when you type an opening parenthesis, which is the method parameter indicator character.

To display the method tooltip, you can also use the key combination `CTRL + SHIFT + SPACEBAR` whenever the cursor is located within a method name or a parameter of a method call. If multiple methods with the same name but different parameters are available, you can choose the appropriate method signature from the list using the up and down arrow keys. If you use the left and right arrow keys, you can display the definitions of the individual parameters.

The method tooltip is displayed automatically when you type the parameters of a method. The method parameter description moves to the next parameter whenever you type a comma (,) before the closing parenthesis.

Comments

The syntax supports single-line and multiple-line (block) comments. A single-line comment starts with two slash marks (`//`) and spans to the end of the current line. The block comment, which can span multiple lines, starts with a slash mark and an asterisk (`/*`) and ends with an asterisk and a slash mark (`*/`).

Single-line comment:

```
// <comment>
```

Multi-line comment:

```
/* <comment> */
```

Syntax

Simple and complex statements are supported. Complex statements usually contain a code body which is opened and closed by braces `{ }`. The body, in turn, can contain any type of statement, that is, the statements can be nested. Simple statements, in contrast, can only be assignments and path expressions, in addition to variable declarations and the import statement.

List of Methods and Keywords

The methods and keywords are divided into the following sections:

- **Contexts**
  Contains information about the `this` keyword.
  - This [page 201]

- **Simple Statements**
  Contains information about statements such as the `import` keyword and the declaration of variables.
  - Import [page 191]
  - Declarations [page 188]
  - Assignment [page 182]

- **Complex Statements**
  Contains information about control and loop statements.
  - If Control Statement [page 191]
  - Switch Control Statement [page 201]
• Foreach Loop Statement [page 189]
• While Loop Statement [page 202]
• Continue Statement and Break Statement [page 186]

• Expressions
Contains information about all types of expressions, for example, literals, path expressions, and arithmetic expressions.
  • Literals [page 192]
  • Function Literals [page 190]
  • Logical Expressions [page 193]
  • Conditional Expressions [page 186]
  • Arithmetic Expressions [page 182]
  • Path Expressions [page 194]
  • Name Qualification [page 193]
  • Action Execution [page 181]

• Association Handling
Contains information about accessing, setting, and checking associations.
  • Association Access [page 183]
  • Set Associations [page 200]
  • Check Associations (IsSet) [page 184]
  • Reset Associations [page 198]

• Business Object Lifecycle
Describes how the lifecycle of a business object instance can be controlled by the coding, that is, when creating, reading, updating and deleting a business object. For more information, see Business Object Lifecycle [page 183].
  • Create Instance [page 187]
  • Create with Reference Action [page 187]
  • Query Execution [page 195]
  • Retrieve Instance [page 198]
  • Delete Instance [page 188]
  • Check Node Existence (IsSet) [page 184]

• Collection Handling
Contains information about how to define and modify collections. For more information, see Collection Handling [page 185].
  • Collection Definition [page 184]
  • Add to Collection [page 181]
  • Remove from Collection [page 197]
  • Clear Collection [page 184]
  • Sort Collection [page 200]
  • Filter Collection [page 189]
  • Count Rows of Collection [page 186]
  • Remove Duplicate Rows from Collection [page 197]
  • Get First or Last Row of Collection [page 191]

• Message Handling
Contains information about how to create and raise messages in the implementation phase.
You cannot receive and handle messages that have been raised by a different script file containing implementation code or the business object infrastructure, for example, during the creation of a business object node instance.

- **Service Integration Methods**
  Contains information about indicators and return values used for condition evaluation in service integrations scenarios. For more information, see [Service Integration Methods](#).
  - **Script File Parameter** [page 199]
  - **Return Values** [page 199]
  - **Process Context** [page 195]

### See Also
- **Actions, Events and Validations** [page 301]
- **Syntax for Business Object Definitions** [page 158]
- **Syntax for Business Object Extension Definitions** [page 170]

#### 5.2.4.2 Action Execution (Business Logic)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Static execution:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;business object name&gt;.&lt;action name&gt;([@param])*</code>;</td>
<td>Instance-based execution:</td>
</tr>
<tr>
<td><code>&lt;node instance var&gt;.&lt;action name&gt;([@param])*</code>;</td>
<td>Description</td>
</tr>
<tr>
<td>An action execution is a special form of a path expression. The code completion offers actions that are based on business object nodes in the same way as common method calls. You can enter parameters in parentheses (...). In reuse library method calls, you can also use optional parameters but not in actions that are based on business object nodes. Even if there are no parameters, you must always use parentheses. In contrast to reuse service methods, action methods based on business object nodes do not support return values. If the multiplicity of the action definition in the business object model is <code>[0..0]</code>, you can execute the action statically by using static path expressions that identify a business object node. If the multiplicity is <code>[1..1]</code> or <code>[1..n]</code>, you can execute the action instance-based. In this case, the instance identifier has to be a collection.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>Instance-based execution ([1:1]):</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>import AP.CRM.Global;</code></td>
<td><code>foreach (var so in SalesOrder.Header.Execute()) {</code></td>
</tr>
<tr>
<td><code>so.Item.CheckATP();</code></td>
<td><code>break;</code></td>
</tr>
<tr>
<td><code>}</code></td>
<td>Static execution ([0..0]):</td>
</tr>
<tr>
<td><code>BOName.NodeName.ActionName();</code></td>
<td></td>
</tr>
</tbody>
</table>

See also [Path Expressions](#), [Import](#), [Create With Reference Action](#).

#### 5.2.4.3 Add to Collection (Business Logic)

| Syntax | `<collection>.Add(<node instance or collection>);` |
Use the `Add` method to add a single row or an existing collection to the end of a collection.

This method cannot be applied to read-only collections. This is only relevant for code related to form data type extensions and in case of mass-enabled script files, in which the `this` keyword represents an immutable collection of nodes.

```javascript
var collection : collectionof Opportunity;
var dataOpportunity : elementsof Opportunity;
var nodeRef;
dataOpportunity.Name = "New opportunity 4711"
nodeRef = this.Create(dataOpportunity);
collection.Add(nodeRef);
```

See also Path Expressions [page 194], Declarations [page 188], Assignment [page 182], Query Execution [page 195]

### 5.2.4.4 Arithmetic Expressions (Business Logic)

**Syntax**

```
literal | <varName> | <path expression>
[ + | — | * | / | % ] <arithmetic expression>;
```

**Description**
The arithmetic expressions support the common mathematical operators for addition, subtraction, multiplication, division, and modulo calculations. Operands can be atomic expressions, such as literals, variable identifiers, path expressions, or other arithmetic expressions.

You can overwrite the precedence order of the operators by using parentheses.

The operands have to evaluate to the exactly same type. The compiler does not implicitly convert incompatible types. The plus sign (+) operator is overloaded to allow string concatenation.

```javascript
var result = (-0.234e-5 + this.Total) * currentValue;
var resultString = "Hello " + "world!";
```

See also Logical Expressions [page 193], Conditional Expressions [page 186], Path Expressions [page 194]

### 5.2.4.5 Assignment (Business Logic)

**Syntax**

```
<x> = <y>;
```

**Description**

An assignment assigns the value of the expression on the right to the expression on the left. The expression on the left must be a variable or an element of a business object instance whereas the expression on the right can be any kind of expression. For the expressions on the right you can use static path expressions, for example, if you want to keep a copy of a reference to static query metadata. The default assignment uses copy semantics.

You cannot use chained assignments, that is, `a = b = c`.

You can also assign node instances on the right to associations on the left. The node instances on the right can either be a newly created node instance or a node instance that has been retrieved through a query. However, this kind of assignment is only possible for modeled associations with [1..1] multiplicity. Node instances to [1..n] associations, which are implemented and not modeled, are assigned implicitly when new node instances are created through create-enabled associations.

```javascript
name = this.Name;
```

See also Contexts, Path Expressions [page 194], Create [page 187]
5.2.4.6 Association Access (Business Logic)

<table>
<thead>
<tr>
<th>Syntax</th>
<th><code>&lt;root entry&gt;..&lt;association name&gt;[.&lt;path expression&gt;]</code>;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The association access is a special form of the path expression. Associations can be defined to be optional and the multiplicity can be [0..1], [1..1] or [1..n]. By following an association, you are able to read data that has been modified but not yet been stored in the database. There are the following categories of associations: Compositions, intra-business-object associations, cross-business-object associations and cross-deployment-unit associations. Compositions are automatically generated associations implemented to allow navigation of the business object node structure (that is, <code>ToParent</code>, <code>ToRoot</code>, navigation to subnodes). Due to the generated association implementation, compositions support the [1..n] multiplicity. Intra-business-object associations link nodes within one business object. Cross-business-object associations link nodes of separate business objects. Intra and cross-business-object associations can only have the [1..1] multiplicity. To exchange data between business objects that are located in different deployment units, configure an internal communication by using the Service Integration Wizard. For more information, see Configure Internal Communication [page 418]. When accessing a [1..n] composition, a collection of nodes is returned. Therefore, you cannot access a member of structured data types of the foreign business object nodes. Instead, you must use the <code>foreach</code> loop statement to access the individual instances in the returned node collection (see the second example below). If the operand on the left of an assignment is an association access path expression followed by an element, data can also be written to the foreign business object node instance. If the node type matches the target definition of the association (see the third example below), you can associate a foreign business object node instance that was just created.</td>
</tr>
<tr>
<td>Example</td>
<td>• Intra or cross-business-object association write access ([1..1] multiplicity): <code>this.AssocToOtherBONode.Value = 42;</code> • Composition association access ([1..n] multiplicity). <code>foreach (var item in this.Items) {</code> <code>totalValue = totalValue + item.ItemValue;</code> <code>}</code> • Associate node instance to current node ([1..1] multiplicity): <code>this.AssocToBONode = &lt;node instance&gt;;</code> The node instance, for example, is retrieved by a query.</td>
</tr>
<tr>
<td>See also</td>
<td>Path Expressions [page 194], Setting Associations [page 200], Checking Associations [page 184]</td>
</tr>
</tbody>
</table>

5.2.4.7 Business Object Lifecycle (Business Logic)

Reading a business object node instances can be performed by either following an association or executing a query. Updates are simply handled by assignment statements; the path expression on the left points to an element of a business object node instance. The changes are automatically committed to the context business object once the script file completes the execution without errors.

The following methods are available for the business object lifecycle:

- Create Instance [page 187]
- Create with Reference Action [page 187]
- Query Execution [page 195]
- Retrieve Instance [page 198]
- Delete Instance [page 188]
- Check Node Existence(IsSet) [page 184]
### 5.2.4.8 Check Associations (Business Logic)

**Syntax**

```plaintext
<path expression>.<association name>.IsSet();
```

**Description**

You can use the `IsSet()` method for `[1..1]` or `[0..1]` associations to check whether the association was set before. Otherwise any access to an association that was not set causes a runtime error.

**Example**

```plaintext
var targetNode = this.ToTarget;
if (this.ToTarget.IsSet()) {
  this.Name = "Assoc set to: " + targetNode.Name;
} else {
  this.Name = "Assoc not set.";
}
```

**See also**

Path Expressions [page 194], Association Access [page 183], Setting Associations [page 200], Resetting Associations [page 198]

### 5.2.4.9 Check Node Existence (Business Object)

**Syntax**

```plaintext
<node instance>.IsSet();
```

**Description**

This method is analogous to the `IsSet` method for checking associations. However, the `IsSet` method for checking node existence checks the node references themselves. You can use this method to check if a new node instance was successfully created.

**Example**

```plaintext
var newBO = BOName.Create();
if (!newBO.IsSet()) {
  raise CreationFailed.Create("E");
} else { … }
```

**See also**

Path Expressions [page 194], Create [page 187], Raising Messages [page 196]

### 5.2.4.10 Clear Collection (Business Logic)

**Syntax**

```plaintext
<collection>.Clear();
```

**Description**

Use the `Clear` method to remove all rows from a collection.

- This method cannot be applied to read-only collections. This is relevant for code related to form data type extensions and in case of mass-enabled script files, in which the `this` keyword represents an immutable collection of nodes.

**Example**

```plaintext
var coll : collectionof Opportunity.Item;
coll.Clear();
```

**See also**

Filter Collection [page 189], Path Expressions [page 194], Declarations [page 188]

### 5.2.4.11 Collection Definition (Business Logic)

**Syntax**

```plaintext
var <variable name> : collectionof <static path expression> | <data type>;
```
If you define a collection explicitly, the collection's row type is inferred from the static path expression, for example, a business object node, or the row type is based on an unstructured data type.

Example

```java
var collection : collectionof ExampleBO.Status;
var collection : collectionof elementsof Opportunity.Item
var collection : collectionof LANGUAGEINDEPENDENT_Text;
//To use the "LANGUAGEINDEPENDENT_Text" GDT, you need to import the AP.common.GDT namespace first
```

See also

Path Expressions [page 194], Declarations [page 188], Assignment [page 182], Query Execution [page 195]

5.2.4.12 Collection Handling (Business Logic)

A collection is a standard table that is defined on the basis of business object nodes or node elements, for example, a collection of opportunities. Collections are generated when instances of business object nodes are returned by the system. You can obtain a collection of business object nodes in several ways, for example:

- You follow [1..n] composition associations. For more information, see Association Access [page 183].
- You execute a Create with reference action with a collection parameter. For more information, see Create with Reference Action [page 187].
- You execute a business object node query. For more information, see Query Execution [page 195].

Besides collections generated by the system, you can also define a collection explicitly. For more information, see Collection Definition [page 184].

The collection handling methods generally apply their logic to a copy of the collection and return a modified copy of the original collection. Therefore these methods declare a collection as return value. However, the Add method and the Clear method behave in a different way. They modify the collection to which they are applied. These methods do not have any return value.

> If you use a method to add or clear a collection or a row in a collection, this only has an impact on the collection itself. To delete or create new business object nodes, use the Delete or Create operation for business objects. For more information, see Business Object Lifecycle [page 183].

The following methods are available for collections:

- Collection Definition [page 184]
- Add to Collection [page 181]
- Clear Collection [page 184]
- Sort Collection [page 200]
- Filter Collection [page 189]
- Remove from Collection [page 197]
- Remove Duplicate Rows from Collection [page 197]
- Get First or Last Row of Collection [page 191]
- Count Rows of Collection [page 186]
5.2.4.13 Conditional Expressions (Business Logic)

Syntax

| Conditional expressions | == | != | < | > | <= | >= | <arithmetical expression> |

Description

Conditional expressions evaluate Boolean values. The supported operators are shown in the syntax above. Use arithmetic expressions as operands, which can also be atomic expressions, such as literals, variable identifiers, and path expressions.

Example

```java
if (fulfillment >= 0.8 && fulfillment < 1.0) { ... }
```

See also Logical Expressions [page 193], Arithmetic Expressions [page 182], Path Expressions [page 194]

5.2.4.14 Continue Statement and Break Statement (Business Logic)

Syntax

<table>
<thead>
<tr>
<th>&lt;loop statement&gt;</th>
<th>{</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;statement&gt;*</td>
<td>continue;</td>
</tr>
<tr>
<td>&lt;complex statement&gt;</td>
<td>{ [ continue;</td>
</tr>
<tr>
<td>}*</td>
<td></td>
</tr>
</tbody>
</table>

Description

The `continue` statement and the `break` statement can be used within the code block of loop statements, such as `foreach` and `while` statements. The `break` statement immediately exits the enclosing loop statement. The `continue` statement quits the current loop iteration and continues with the next iteration at the top of the code block of the enclosing loop statement. In nested loop statements, however, the `break` statement only exits the inner loop. Usually you need to embed the `continue` statement or the `break` statement into an enclosing `if` condition in order to avoid endless loops or a forced single loop execution.

Example

```java
foreach (node in query.Execute()) {
    count = count + 1;
    if (count < 3) continue;
    if (count > 5) break;
    node.SomeAction();
}
```

See also Foreach Loop Statement [page 189], While Loop Statement [page 202]

5.2.4.15 Count Rows (Business Logic)

Syntax

| <intVar> = <collection>.Count(); |

Description

If you use the `Count` method, the system returns the number of rows in a collection.

Example

```java
var coll : collectionof SalesOrder.Status;
var counter;
counter = coll.Count();
```

See also Path Expressions [page 194], Declarations [page 188], Assignment [page 182]
### 5.2.4.16 Create Instance (Business Logic)

**Syntax**

- Static creation of business object (root node):
  
  <business object name>.Create({},<node data>);

- Node creation through association that is create-enabled:
  
  <this | NodeInstanceVar>.<association name>.Create(<node data>);

**Description**

You can use the `Create` method for the following:

- Use the method to create complete business objects (that is, the root nodes) in a static manner.
- You can also use the method to create sub-nodes of existing business objects, for example, through composition access because this access implicitly maintains the association relationship between the parent and root node. The `Create` method binds to static member path expressions that identify a business object node. It also binds to associations that are create-enabled, for example, compositions or intra-business-object associations. In compositions, the parent node instance is automatically derived from the preceding path expression and is used for the composition maintenance.
  
  The instance-based `Create` method for sub-node creation is not available for variables that are derived from association access because in this case the parent node cannot be reliably determined.
- Optionally, you can also provide node data to the `Create` method as a parameter. When provided, this data is automatically copied to the newly created node instance. The parameter type is the complete node element structure. You can infer the node element type as part of the variable declaration by identifying a certain node through a static path expression. For more information, see Declarations [page 188].

You can also assign the data to the node instance after the node has been created. However, if you create instances of SAP business objects, for some of the business objects you must provide default values in the parameters at the time the business objects are created.

**Example**

```java
var newBO = TestBO.Create();
var itemData : elementsof TestBO.Items;
var newItem;
itemData.ItemName = "Test";
itemData.ItemValue = 42;
newItem = this.Items.Create(itemData);
```

**See also**

Path Expressions [page 194], Import [page 191], Create With Reference Action [page 187]

### 5.2.4.17 Create With Reference Action (Business Logic)

**Syntax**

```java
<business object name>({<node name>}).<CWR action name>({SomeNode|NodeCollection})(<param>)*);
```

**Description**

The `Create with reference` (CWR) action is a special type of action that can only be executed statically and that directly returns one or more instances of the node to which the action belongs. All CWR actions are overridden to offer two signatures, one with a single node parameter and one with a collection of nodes. CWR actions return a single node or a collection of nodes depending on the collection property of the method call parameter.

CWR actions can only be used by SAP business objects.

**Example**

```java
import AP.CRM.Global;
var salesOrders = SalesOrder.QueryByElements.Execute();
var allSOSCopied = SalesOrder.Copy(salesOrders);
var singleCopiedSO;
foreach (var so in salesOrders) {
    singleCopiedSO = SalesOrder.Copy(so);
    break;
}
```
5.2.4.18 Declarations (Business Logic)

**Syntax**

```plaintext
var <variable name> [: collectionof? elementsof? <static path expression> | <data type>]?{= literal | <path expression>}?;
```

**Description**
The implementation language is a statically-typed language that allows you to use declared variables only. You always need to declare the variable before you can use it in a statement. If you declare a variable within a code block (so-called variable scoping), that is, within the braces ({}), the scope of this variable is limited to this code block.

Variable shadowing is not supported, that is, variables cannot reuse names that are already declared in an enclosing scope.

The variable type is automatically inferred upon first assignment of the variable. You can assign an initial value either by specifying a literal or by specifying a path expression, that is, elements of other already existing variables or elements of the `this` context. Assignments use copy semantics, that is, a temporary copy is created when individual elements are accessed using the `this` context. Node variables have reference semantics. Alternatively, you can explicitly specify the variable type in the variable declaration. For this, you can use basic data types as well as global data types (GDTs).

You can also derive the underlying type of a business object node by using the `elementsof` modifier followed by a static path expression that identifies a business object node. This is useful if you define initial node data that can be passed as a parameter to the `Create` operation of a business object node. Do not use the `elementsof` modifier if you want to derive the type of an element from a static path expression.

**Example**

```plaintext
var counter = 0;
var amount : Amount;
//To use the "Amount" GDT, you need to import the AP.common.GDT namespace first
var strVar = "test";
var strVar : LANGUAGEINDEPENDENT_Text;
//To use the "LANGUAGEINDEPENDENT_Text" GDT, you need to import the AP.common.GDT namespace first
var boolVar = true;
var name = this.Name;
var nodeData : elementsof ExampleBO.Item;
var nodeData : ExampleBO.Item.Element;
var coll : collectionof ExampleBO.Item;
var coll : collectionof elementsof ExampleBO.Item;
```

**See also**

Path Expressions [page 194], Import [page 191], Action Execution [page 181]

5.2.4.19 Delete Instance (Business Logic)

**Syntax**

```plaintext
<this | NodeInstanceVar>.Delete();
```

**Description**
The `Delete` operation, which uses the inverse semantics of the `Create` operation, can only be called in the instance-based variant and must be applied to business object nodes. The reason is that the instance to be deleted must be identified by the preceding member path expression. The `Delete` operation can also be applied to collections of business object nodes.
Example

```csharp
foreach (var BO in TestBO.QueryByElements.Execute()) {
    if (BO.Name == "delete") {
        BO.Delete();
    }
}
```

Deletion of a collection of nodes:
```csharp
TestBO.QueryByElements.Execute().Delete()
```

See also

Path Expressions [page 194], Import [page 191]

### 5.2.4.20 Filter Collection (Business Logic)

**Syntax**

```csharp
<collection A> = <collection B>.Where(<logical expression>);
```

**Description**

You can use the `Where` method to filter a collection in order to return only the rows that meet the criteria. This method returns a new collection, for example, collection A, which contains a subset of the source collection, for example, collection B.

**Example**

In the following example, the collection `collB` is the result of a query that was executed previously:
```csharp
var mySearchValues : elementsOf myBO.Item;
var collA;
mySearchValues.ItemID.content = "Bill";
collA = collB.Where(n => n.ItemID != mySearchValues.ItemID || (!
(n.ItemID.content < 1000 && n.ItemID.content == "Fred"));
var collStrings : collectionOf LANGUAGEINDEPENDENT_Text;
//To use the "LANGUAGEINDEPENDENT_Text" GDT , you need to import the
AP.common.GDT namespace first
var collStringsResult;
collStringsResult = collStrings.Where( n => n == "Fred" );
collStringsResult = collStrings.Where( n => n.StartsWith("Fred") );
```

See also

Clear Collection [page 184], Logical Expressions [page 193], Function Literals [page 190], Path Expressions [page 194], Declarations [page 188], Assignment [page 182], Query Execution [page 195]

### 5.2.4.21 Foreach Loop Statement (Business Logic)

**Syntax**

```csharp
foreach (var? <identifier> in <path expression>) {
    <statement list>
}
```
The `foreach` loop statement allows you to iterate over collections of any type. Variables that are used in a `foreach` statement are called loop variables. They have different semantics depending on the way they are declared:

- Declaration of loop variable before the `foreach` statement:
  If a loop variable is declared before the `foreach` statement, the variable type must match the row type of the collection. In this case, the loop variable is defined in the scope outside of the `foreach` statement and can therefore also be accessed after the `foreach` statement. If the loop variable refers to a collection of data types, the variable has value semantics, that is, it is a copy of the original data of the collection element in the current iteration. Therefore the original data in the collection cannot be changed. If a loop variable refers to a collection of business object nodes, the variable has reference semantics, that is, you can always change the data of the collection elements.

- Explicit declaration of loop variable as part of the `foreach` statement:
  If you declare a loop variable within the `foreach` statement using the `var` keyword, the scope of the variable is limited to this code block, that is, to the code within the braces of the statement. Therefore the data of the last loop iteration cannot be accessed via the variable outside of the `foreach` code block. A loop variable that is declared explicitly within the `foreach` statement always has reference semantics, even if the variable refers to a collection of data types. This means that the original data of the collection elements can be changed. This is particularly useful for form data type extensions in the following case: a data type with a sub-element that is a collection is passed in as the `FormData` parameter and you want to change existing data in the collection elements. The loop variable itself, however, is immutable.

The statements within the `foreach` body can make use of the loop variable to access the instance of the current iteration. For information about how to obtain collections of business object nodes, see Collection Handling [page 185].

**Example**

```javascript
var node;
foreach (node in query.Execute()) {
  count = count + 1;
  if (count < 3) continue;
  if (count > 5) break;
  node.SomeAction();
}
```

Example for form data type extension:

```javascript
foreach (var elemData in FormData.CollectionSubElement) {
  elemData.Element = "New Value";
}
```

**See also** Path Expressions [page 194], Query Execution [page 195], Continue Statement and Break Statement [page 186]

### 5.2.4.22 Function Literals (Business Logic)

**Syntax**

Applicable in collection handling when you use the `OrderBy` and `DistinctBy` methods:

```javascript
<function parameter name> => <function parameter name>.<method or property of inferred type>
```

Applicable in collection handling when you use the `Where` and `Remove` methods:

```javascript
<function parameter name> => <function parameter name>.<method or property of inferred type> <arithmetic expression>
```

**Description**

You can only use function literals in the following collection handling methods: `Where` [page 189], `OrderBy` [page 200], `DistinctBy` [page 197], and `Remove` [page 197]. A function literal is an expression that defines an unnamed function. Function literals use the lambda operator (`=>`), which reads as “goes to”. The left side of the lambda operator specifies the input parameter, the type of which is inferred from the last item of the path expression. The right side of the lambda operator holds the statement. In `OrderBy` methods and `DistinctBy` methods, the statement is a path expression and in `Where` methods and `Remove` methods the statement is a comparison. In all these cases, the path expression must specify a structured element.

- Do not use a function parameter name that you have already used for a declared variable. For more information, see Declarations. [page 188].
**Example**

```csharp
var myCollection : collectionof TargetBO;
myCollection = this.ToNAssoc.Where(n => n.elementX == "Yes");

var myCollection : collectionof TargetBO;
foreach (var i in this.ToNAssoc) {
    if (i.elementX == "Yes") { myCollection.Add(i); }
}
```

See also [Declarations](#), [Foreach Loop Statement](#), [Collection Handling](#)

---

### 5.2.4.23 Get First or Last Row (Business Logic)

**Syntax**

```
<collection row> = <collection>.GetFirst();
<collection row> = <collection>.GetLast();
```

**Description**

Use the `GetFirst` or `GetLast` method to return the first or last row of a collection.

**Example**

```csharp
var coll : collectionof Opportunity.Item;
var first;
var last;
first = collection.GetFirst();
last = collection.GetLast();
```

See also [Path Expressions](#), [Declarations](#), [Assignment](#)

---

### 5.2.4.24 If Control Statement (Business Logic)

**Syntax**

```
if (<logical expression>) {<statement list>}
[ else if (<logical expression >) {<statement list>} ]*
[ else {<statement list>} ]?
```

**Description**

The `if` statement is a control statement. Depending on the result of the logical expression, the system executes or skips the statement block. Optionally, `else if` and `else` subclauses can be used.

**Example**

```csharp
Bonus calculation:
if (fulfillment >= 0.8 && fulfillment < 1.0) {
    bonus = this.Actual.content * 0.1;
} else if (fulfillment >= 1.0) {
    bonus = this.Actual.content * 0.15;
} else { bonus = 0; }
```

See also [Logical Expressions](#)

---

### 5.2.4.25 Import (Business Logic)

**Syntax**

```
import <namespace component>[.[<namespace component>]]*;
```
The `import` keyword imports business objects or reuse libraries from namespaces that are available in the SDK in the Repository View. Imports have to be defined at the very top of a source file. You can only import complete namespaces but no individual entities. If there are naming conflicts between different entity types in the same namespace, for example, if a reuse library has the same name as a business object, you must fully qualify the name upon usage.

The namespace syntax in the scripting language is different to the namespace representation in the Repository View. In the scripting language, the constant `http://sap.com/xi` prefix of the URL-based namespaces is omitted and the slash mark (`/`) is replaced by a period (`.`).

Imported business objects can be statically accessed by typing the business object name as the root identifier of a path expression. The local namespace of the solution is imported implicitly, that is, local business objects can be accessed without an `import` statement.

If you use more than one namespace, we recommend that you define an alias for each imported namespace and qualify business objects or data types from these namespaces by using the aliases. For examples, see Name Qualification [page 193].

Code completion is available for the namespace import, as well as for the static access of imported business objects and reuse libraries. The code completion for the namespace import always offers the required amount of subsequent path elements in order to complete the namespace so that the corresponding entities can be accessed, for example, business objects or reuse libraries.

**Example**
```
import ABSL;
import AP.CRM.Global as CRM;
```

**See also** Path Expressions [page 194]

---

### 5.2.4.26 Literals (Business Logic)

**Syntax**
```
"<literal string>" | true | false |
[+|-]? DecDigit* .? DecDigit+ [ e|E [+|-]? DecDigit ]?;
```

**Description**
The following literals can be used:
- Literal strings that are included in quotation marks (`".."`)  
- Numeric literals, that is, positive and negative integers and floating points  
- The Boolean values `true` and `false`

You can use a literal, for example, as an expression on the right of an assignment or as a parameter of a method call in a path expression. In a literal string, you can use a backslash (`\`) to escape characters, such as carriage returns or quotation marks (`"`). You can use the following escape sequences:
- "\n" (new line)  
- "\r" (carriage return)  
- "\t" (tab)  
- "\v" (vertical tab)  
- "\f" (form feed)  
- "\" (backslash)  
- "\" (quotation marks)

**Example**
```
var counter = 0;
var strVar = "test";
var escapedStringVar = "FirstLine\\nSecondLine";
this.ActionName(true);
```

**See also** Assignment [page 182], Path Expressions [page 194]
5.2.4.27 Logical Expressions (Business Logic)

| Syntax | ![]? <conditional expression> [ && | || ![]? <conditional expression> ]* |
|---|---|

**Description**
The logical operators **AND** (&&) and **OR** (||) always evaluate the overall expression to a Boolean value. The operands themselves are conditional expressions, which in turn are comprised of arithmetic expressions, finally breaking down to atomic expressions, such as literals, variable identifiers, and path expressions. You can also use the **NOT** operator (!) combined with brackets to override the precedence default value of subexpressions.

Short circuit evaluations are not supported. That is, if in a logical AND or OR expression the value of the first operand is sufficient to determine the result of the operation, the second operand is still evaluated.

**Example**
```java
if (fulfillment >= 0.8 && fulfillment < 1.0) { ... }
check = !(this.Element > 42);
```

**See also**
Conditional Expressions [page 186], Arithmetic Expressions [page 182], Path Expressions [page 194]

5.2.4.28 Name Qualification (Business Logic)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>&lt;object type 1&gt;::&lt;alias for namespace&gt;::&lt;path element&gt;::&lt;object type 2&gt;::&lt;path element&gt;</th>
</tr>
</thead>
</table>

**Description**
Usually, you can address the imported objects directly by their names instead of using fully qualified names. However, if more than one object type (for example, a business object node, query, association, or action) is declared with the same name, you need to fully qualify the objects in order to make them unambiguous. This is necessary, for example, in the following cases:
- Two namespaces are imported that contain business objects with the same name.
- You have created a business object with the same name as an SAP business object. To reference the SAP business object, you need to enter the fully qualified name of this business object including its namespace alias. Without mentioning the namespace, the system will choose your business object instead of the SAP business object.
- You want to access a business object that has the same name as a reuse service or a message.
- A node name in one business object is the same as a query name in the same business object on the same level.

If the system discovers such a naming conflict, you can either use code completion to insert the correct path expression or enter it directly by using two colons (::).
Example

Fully qualifying names because two namespaces contain business objects with the same name:

```javascript
import AP.CRM.Global as CRM;
var sapSalesOrder = CRM:SalesOrder;
var mySalesOrder = SalesOrder;
```

Fully qualifying names because there are nodes, elements or associations within the same name:

```javascript
import ABSL;
import AP.FinancialAccounting.Global as fin;
var costObject = fin:FinancialAccountingViewOfCostObject;
var descriptionNodeElements = elementsOf
  fin:FinancialAccountingViewOfCostObject.Node::Description;
var description = 
  fin:FinancialAccountingViewOfCostObject.Element::Description;
```

costObject = fin:FinancialAccountingViewOfCostObject.Create();
descriptionNodeElements.Description.content = "A Description";
descriptionNodeElements.Description.languageCode = LanguageCode.ParseFromString("EN");
costObject.Association::Description.Create(descriptionNodeElements);
description = costObject.Element::Description;

See also Path Expressions [page 194], Declarations [page 188], Import [page 191], Association Access [page 183], Query Execution [page 195]

5.2.4.29 Path Expressions (Business Logic)

Syntax

<root item>[.<data type element> | business object node feature | method name

[ ( <param>* ) ]]*

Path expression containing a library function parameter (for example, of a reuse library service) for use in a variable declaration:

```<library>.<library function>.<param>...```.

Description

Path expressions can be used for the following:

- Node elements
- Associations
- Method calls
- Type of variables

Path expressions always start with a root item. The root item can be a variable identifier, the this context, a static access to an imported business object or a reuse library, as well as a message declared in the business object definition.

If the root item is a variable of a structured data type, the path expression addresses subelements of this data type. If the this context and imported business objects are used, all supported business object node features can be addressed through the path expressions. The business object node features are element and subelement access, associations, queries, actions, including Create with reference [page 187] actions, and the Create [page 187] operation and the Delete [page 188] operation. If the root item is a reuse library or a Web service (library), you must add the reuse service or Web service operation (library function) and at least the parameter. If the parameter is of a structured type, the path expression can also include the elements and subelements of this type.

Associations are instance-based and they can therefore only be followed by the this context and by variables that are typed as business object nodes. In contrast, queries are static entities that have to be retrieved and executed based on imported business objects. As an action can have the multiplicity [0..0], [1..1], or [1..n], a query can either be accessed statically or instance-based, which depends on the multiplicity. For the execution of an action, you can provide parameters to the action method call in the same way as you can provide parameters for the execution of any reuse library service.

Methods and complete reuse libraries can be bound to certain data types and are then automatically offered in the code completion. In addition to actions that are defined as part of the business object model and that belong to a certain business object node, there are also various methods that are available for all unstructured data types, for example, the ToString method.

Example

```javascript
var product = this.ProductName;
var y : CurrencyConversion.Add.TargetCurrencyCode;
```
5.2.4.30 Process Context (Business Logic)

**Syntax**

```
<path expression to node>.GetProcessContext().<process context field>;
```

**Description**

Within an `.abs1` file related to service integration, you can use the `GetProcessContext` method for every accessible node instance. The process context itself offers change notifications for all changed reference business object instances and all already evaluated conditions of the reference business object root node instances.

For the root node of the reference business object, the process context provides the Boolean field `IsActiveInProcess`.

The following change notification fields are available for all accessible nodes (including nodes that are accessed through associations):

- `IsCreated`
- `IsUpdated`
- `IsDeleted`

The following evaluated conditions are available for the root node of the reference business object:

- `IsStarted`
- `IsChanged`
- `IsCancelled`

These conditions return the default condition evaluation result for the root node of the reference business object.

**Example**

```
var isActive = this.SubNode.GetProcessContext().IsActiveInProcess;
```

See also [Script File Parameters](#), [Return Values](#)

5.2.4.31 Query Execution (Business Logic)

**Syntax**

```
<query>.Execute ([[<selection parameter collection>]]?);
```
You can execute queries based on business object nodes only. These queries are defined and bound to certain business object nodes. For more information, see Queries [page 302].

Queries read the data of a node directly from the database and do not take into account data that has been modified but not yet stored in the database.

The query execution always returns only those instances of the nodes, to which the query is bound. However, the optional query parameter structure can also contain elements that are not part of a node’s data type element structure. Queries must be statically accessed once the business object that hosts the node with the desired query is imported.

Queries can directly be executed if no further query parameters need to be specified. Usually, you need query parameters to narrow down the result to the desired amount by defining appropriate filtering criteria. You can create an empty selection parameter collection by using the CreateSelectionParams() method of the static query metadata. You can add instances of selection parameters to the collection by using overloaded versions of the Add method.

The query selection parameter collection is structured as follows:

- The first argument of the Add method identifies the element of the query parameter structure, for which you can define a filter criterion. To specify the query parameter structure element you must use a path expression that is based on the query which was used to create the select options. The element identified in the query parameter structure must not be structured.
- The second element defines the so-called “sign” with the potential values "I" (inclusive) and "E" (exclusive).
- The third parameter defines the so-called “option” with potential values, such as "EQ" (equals) and "CP" (contains pattern).
- The last two parameters "low" and "high" define the actual filter values. The low parameter is optional and can be used for “options” that allow for ranges.

When you use the method tooltip, the values for the parameters of the Add method, such as “sign” and “option”, are automatically displayed.

The query metadata also includes association names, which can be used to query all instances that reference a certain associated node instance. In this case, a special Add method signature with only two parameters is supplied, of which the first parameter has to be an association of the query metadata and the second parameter has to be a node instance of the expected association target type.

### Example

```javascript
import <namespace>;
var queryResult;
• Schematic example:
  var query = <business object name>[.<node name>].<query name>;
  var selParams = query.CreateSelectionParams();
  selParams.Add(query.<element>, "I", "EQ", this.<element>);
  queryResult = query.Execute(selParams);

• Concrete example:
  var query = SalesOrder.Header;
  var selParams = query.CreateSelectionParams();
  selParams.Add(query.PostingDateTime, "I", "GT", this.DateFrom);
  queryResult = query.Execute(selParams);

• Example for association selection (find all other business object instances that reference “this”):
  var query = OtherBO.QueryByElements;
  var selParams = query.CreateSelectionParams();
  selParams.Add(query.ToSomeMyBO, this);
  queryResult = query.Execute(selParams);
```

See also Path Expressions [page 194], Import [page 191], Collection Handling [page 185]

### 5.2.4.32 Raise Message (Business Logic)

#### Syntax

```javascript
raise <message name>.Create(<severity> [, <message text variable>]*);
```
You can use the `raise` keyword to raise messages that have been declared in the business object definition. When you use the `Create` method, an instance of a message is created that is automatically bound to the message type. The `Create` method has one mandatory parameter: the severity. You have to supply the severity as a string:

- "E" for errors
- "W" for warnings
- "S" for success messages

As additional optional parameters you can supply the values for placeholder variables as declared in the message text. The type of these values is `ANY`.

You do not need to import messages if they are declared in the business object definition. They are then automatically available in the code completion of the current context.

Example
```
raise MissingTest.Create("E", this.ObjectID);
```

See also [Path Expressions](#page_194)

### 5.2.4.33 Remove Duplicate Rows (Business Logic)

**Syntax**

```
<collection A> = <collection B>.DistinctBy(<function literal>).ThenBy(<function literal>);
```

**Description**
You can use the `DistinctBy` method to delete duplicate rows in a collection according to the specified criteria.

Example
```
var collMyCampaigns : collectionof Campaign;
var result;
result = collMyCampaigns.DistinctBy( n=>n.Status.LifeCycleStatusCode ).ThenBy( m=>m.PlannedStartDate );
var collStrings : collectionof LANGUAGEINDEPENDENT_Text;
//To use the "LANGUAGEINDEPENDENT_Text" GDT, you need to import the
AP.common.GDT namespace first
var collStringsUnique;
collStringsUnique = collStrings.DistinctBy( n=>n );
```

See also [Declarations](#page_188), [Function Literals](#page_190), [Path Expressions](#page_194), [Assignment](#page_182)

### 5.2.4.34 Remove from Collection (Business Logic)

**Syntax**

```
<collection A> = <collection B>.Remove(<logical expression>);
```

**Description**
You can use the `Remove` method to remove those rows from the collection that meet the criteria. This method returns a new collection, for example, collection `A`, which contains a subset of the source collection, for example, collection `B`.

Example
```
In the following example, the collection `collB` is the result of a query that was executed previously:
var mySearchValues : elementsof myBO.Item;
var collA;
mySearchValues.ItemID.content = "Smith";
collA = collB.Remove(n => n.ItemID != mySearchValues.ItemID &&
n.ItemID.content != "Miller");
```

See also [Logical Expressions](#page_193), [Function Literals](#page_190), [Path Expressions](#page_194), [Declarations](#page_188), [Assignment](#page_182), [Query Execution](#page_195)
5.2.4.35 Reset Associations (Business Object)

Syntax

```
<path expression>.<association name>.Reset();
```

Description

This method allows you to reset an association. If, for example, a target node instance has been deleted by following the association, you must reset the association because the references that point to the deleted business object are not cleared automatically. Therefore, such invalid references must be explicitly deleted and every access to an association must be secured by a previous `IsSet` check call. For information about the `IsSet` check for associations, see [Check Associations](page 184).

The `Reset` method can only be applied to foreign key associations and is not available for variables that are derived from association access.

Example

```javascript
var targetNode = this.ToTarget;
if (targetNode.IsSet()) {
    targetNode.Delete();
    this.ToTarget.Reset();
}
```

See also [Path Expressions](page 194), [Association Access](page 183), [Set Associations](page 198), [Delete](page 188)

5.2.4.36 Retrieve Instance (Business Logic)

Syntax

```
<node instance> = <business object>.Retrieve(<alternative key>);
<collection of node instances> = <business object>.Retrieve(<collection of alternative keys>);
```

Description

You can use the `Retrieve` method to read the data of a single node instance or a collection of node instances of a business object. In contrast to the `Query` method, which always reads data directly from the database, the `Retrieve` method enables you to read data that has been modified but not yet been stored in the database. As parameters for the method, you can pass a single alternative key or a collection of alternative keys, which have been declared as variables. The variables, which contain the alternative key values, need to be of the same type as the alternative keys specified in the business object. If the method is used for a collection of node instances, it returns a new collection.

Example

Definition of alternative key in business object:

```
businessobject BonusPlan {
}
```

Script file:

```javascript
var elementID : xyz.SO_ID;
var collectionID : collectionof xyz.SO_ID;
var retrieveColl;
var retrieveSingle;
elementID.content = "5000000002";
retrieveSingle = SalesOrder.Retrieve(elementID);
collectionID.Add(elementID);
elementID.content = "5000000003";
collectionID.Add(elementID);
retrieveColl = SalesOrder.Retrieve(collectionID);
```

See also [Declarations](page 188), [Collection Handling](page 185)
5.2.4.37 Return Values (Business Logic)

**Syntax**

```
return <literal> | <path expression>;
```

**Description**
The use of the `return` keyword is mandatory in .abs1 files that were created by using the Service Integration wizard. The return value has to be of Boolean type. For each of the four condition evaluations (Relevance, Start, Change, Cancel) that are possible in a service integration scenario, the Boolean return value must be used to signal the calling process agent about the evaluation result. To calculate the condition for the current root node instance, you can access the process context information of this node and its node values.

**Example**

```
return true;
return this.Count > 7;
```

See also [Script File Parameters](#), [Process Context](#)

5.2.4.38 Script File Parameter (Business Logic)

**Description**
The `InReconciliation` Boolean indicator specifies conditions that depend on the mode of execution, that is, the standard message mode or the reconciliation message mode. The reconciliation mode is an error recovery mechanism that allows it to send a new message, the reconciliation message. This reconciliation message contains the complete information and cancels all previous erroneous messages.

**Example**

```
if (InReconciliation) {
    ...
} else 
return false;
```

See also [Return Values](#), [Process Context](#)

5.2.4.39 Service Integration Methods (Business Logic)

Service integration allows you to synchronize data between several involved business objects that are located in different deployment units in an asynchronous way. The business objects exchange the data through XML messages. You can use the methods listed below to determine for which nodes of the sending business object the data is transmitted.

Script files can be used for the condition evaluation, which specifies whether the data of a node is part of the cross-business-object synchronization. Therefore, the .abs1 files in service integration scenarios support parameters that are passed in at runtime. Additionally, a Boolean return value is expected that signals to the calling process agent whether the business object instance is to be considered in the message construction. Furthermore, the process context is attached to every business object node instance that allows you to check certain status information related to service integration.

In script files that are executed within a service integration scenario, the context information is read-only. This means that the system prevents accessible business object nodes and calls of actions from being changed. The information about the script file signature (that is, the parameters and return values and their corresponding types) is documented in a comment block at the very beginning of each .abs1 file. In service integration scenarios, various .abs1 file subtypes are involved. This is also documented in the comment.

The following methods are available for service integration:
5.2.4.40 Set Associations (Business Logic)

Syntax

\[
<\text{path expression}.<\text{association name}> = <\text{path expression}>;
\]

Description

You can navigate along associations that have been set by the user interface but you can also set associated node instances explicitly. For this, you can use a simple standard assignment expression that on the left provides a path expression, whose last element is an association. The path expression on the right must evaluate to the node type that is expected to be the target node of the modeled association. For such assignments, you can only use modeled associations with \([1,1]\) or \([0,1]\) multiplicity. The expression on the right can be, for example, an item of a query execution result or a Create operation.

Example

- Assign node instance retrieved by query execution:
  
  ```java
  foreach (var item in query.execute(selParams)) {
    this.AssocToTarget = item;
    break;
  }
  ```

- Assign newly created node instance:
  
  ```java
  var data : elementsof TargetBO;
  data.Name = "Created";
  this.ToTarget = TargetBO.Create(data);
  ```

See also

Path Expressions [page 194], Association Access [page 183], Create Instance [page 187], Check Associations [page 184]

5.2.4.41 Sort Collection (Business Logic)

Syntax

\[
<\text{collection A}> = <\text{collection B}>.\text{OrderBy}(\text{<function literal>}).\text{ThenByDescending}(\text{<function literal>}).\text{ThenBy} ... ;
\]

\[
<\text{collection A}> = <\text{collection B}>.\text{OrderByDescending}(\text{<function literal>}).\text{ThenBy}(\text{<function literal>}).\text{ThenBy} ... ;
\]

Description

You can use the \text{OrderBy} or \text{OrderByDescending} method to sort a collection in ascending or descending order according to the specified sorting criteria. By default, collection columns of numeric and byte type are sorted by their value and columns of string type are sorted as text. You can define the sorting criteria as action parameters by using a function literal, which refers to the columns of the collection. To define multiple sorting criteria, add \text{ThenBy} or \text{ThenByDescending} statements to the expression.

1. If the row type of a collection is structured, you need to specify the sorting criteria. The sorting parameters must be an atomic data type, that is, you are not allowed to define a structured attribute of the collection row type as a sorting parameter. If the row type is unstructured, you can only sort by row and cannot specify a sorting criteria.

Sorting does not have a permanent effect on business object nodes. Each time a node is read from the system, all previous sorting results are lost.

Example

```java
var collItems : collectiono\text{f} Opportunity.Item;
var collItemsSorted : collectiono\text{f} Opportunity.Item;
collItemsSorted = collItems.OrderBy(n=>n.NetAmount.currencyCode).ThenBy(n=>m.NetAmount.content);
var collStrings : collectiono\text{f} String;
var collStringSorted;
collStringSorted = collStrings.OrderByDescending( n => n );
```
### 5.2.4.42 Switch Control Statement (Business Logic)

**Syntax**

```plaintext
switch ( <expression> ) {
    case <expression>) [, <expression>* ] { <statement list> }
    [ default { <statement list>} ] ?
}
```

**Description**
The `switch` control statement provides a more convenient option to express `if... else if...` code blocks that evaluate exactly the same field against different values again and again. The `switch` type is limited to non-structured data types. The `case` clause allows for arbitrary expressions, for example, the use of variables. The `default` clause is optional.

**Example**

```plaintext
switch (this.Amount.content) {
    case 1, 2 {this.Text = "alt1"; }
    case varName.element {this.Text = "alt2"; }
    default { this.Text = "initial"; }
}
```

**See also**
- Control Statements  [page 191], Path Expressions  [page 194]

### 5.2.4.43 This (Business Logic)

**Syntax**

```plaintext
this.<feature name>;
```

**Description**
The `this` keyword defines the context in which the script file is executed. You must use the keyword as the first member of a path expression that accesses the context. The keyword has the type of the business object node that the script file is assigned to and, at runtime, points to the current instance of such a business object node. The keyword can define the root of a path expression, just as imported business objects do for static access. The `this` keyword itself is immutable and cannot be modified, whereas the elements of the node represented by `this` can be modified.

1. If you use the `this` keyword in the coding of a mass-enabled script file, `this` represents a collection of business object nodes.

The code completion for the business object node specified by `this` displays instance-based features such as node elements, associations and instance-based actions including mass-enabled features. Queries and static actions are not instance-based and therefore you can address them only through static business object access.

You can use the `GetObjectNodeReference()` method along with the `this` keyword to return the following information about the business object nodes in a solution:
- Node ID of the relevant business object node
- ObjectTypeCode of the business object
- ObjectNodeTypeCode of the node

**Example**

```plaintext
var name = this.Name;
var ObjNodeRef = this.Item.GetObjectNodeReference();
```

**See also**
- Path Expressions  [page 194], Assignment  [page 182], Import  [page 191]
5.2.4.44 While Loop Statement (Business Logic)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>while (&lt;logical expression&gt;) { &lt;statement list&gt; }</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The while loop statement allows you to execute the statement list in the code body as long as the logical expression evaluates to true.</td>
</tr>
<tr>
<td>Example</td>
<td>while (this.StrElement.Length() &lt; 10) { this.StrElement = this.StrElement + &quot;Test&quot;; if (this.StrElement.Length() &gt; 7) break; }</td>
</tr>
<tr>
<td>See also</td>
<td>Logical Expressions [page 193], Continue Statement and Break Statement [page 186]</td>
</tr>
</tbody>
</table>

5.2.5 Reuse Libraries

5.2.5.1 Basic Reuse Libraries

5.2.5.1.1 Basic Reuse Libraries Reference

The basic reuse libraries of the scripting language extend the scope and functions that you use to implement your business logic in the SDK. These “language-near” libraries provide basic functions that are used quite often, for example, to retrieve context data, such as the current date or time for the current identity.

Reuse libraries allow for parameters that are either data types (CDTs or GDTs) or business object node instances. The return values have to be data types.

To use the basic reuse libraries, you need to import the namespace ABSL with the import keyword. For more information on the import keyword, see Syntax for Implementation of Business Logic [page 178].

The services of the basic reuse libraries can always be accessed in a static manner. In addition, most of the libraries are bound to a certain data type. In this case, the instance of the data type on which the method is executed will be automatically passed to the method at runtime as the first parameter.

import ABSL;
var globalDateTime = GlobalDateTime.ParseFromString("2008–03–30");
var date = globalDateTime.ConvertToDate():

List of Basic Reuse Libraries

- Context [page 204]  
  Provides context information, for example, the current time in the time zone of the user or the identity UUID of the user that is currently logged on.
- **Duration** [page 211]
  Compares durations and performs simple calculations with durations.

- **Numeric** [page 218]
  Provides basic functions for integers and decimals.

- **Numeric Character** [page 219]
  Converts a string into a numeric character.

- **Currency Conversion** [page 205]
  Provides functions for currency-based amounts. For example, you can convert an amount from one currency into another.

- **Quantity Conversion** [page 219]
  Compares quantities and performs simple calculations with quantities.

- **UUID** [page 223]
  Provides basic functions for UUIDs.

- **Language Code** [page 217]
  Converts a string into a language code.

- **Date** [page 207]
  Compares dates, performs simple calculations with dates, and provides further useful date functions.

- **Time** [page 221]
  Compares times and provides further useful time functions.

- **Date and Time** [page 210]
  Converts a string into a date and time value.

- **Global Date and Time** [page 213]
  Compares global date and time values, performs simple calculations with global date and time values, and provides further useful global date and time functions.

- **Local Date and Time** [page 217]
  Converts local date and time values.

- **Local Normalised Date and Time** [page 218]
  Converts local normalized date and time values.

- **Web Service Utilities** [page 223]
  Calls REST mashup Web services.

**See Also**

- Reuse Libraries for Business Areas  [page 223]
- Built-In Functions  [page 228]

### 5.2.5.1.2 Binary (Reuse Library)

This reuse library contains a service that you can use to convert a string to a binary value.

**Binary.ParseFromString**

<table>
<thead>
<tr>
<th>Description</th>
<th>Converts a string to a binary value using UTF-8 encoding.</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Note]</td>
<td>This service supports static calls only.</td>
</tr>
</tbody>
</table>
Example
```
this.Binary.content = Binary.ParseFromString("This is a test string.");
```

Result: Binary value 546869732069732061207465737420737472696E672E

5.2.5.1.3 Context (Reuse Library)

This reuse library contains services that provide context information, for example, the current time in the time zone of the user or the identity UUID of the user that is currently logged on.

- **The context reuse library supports static calls only.**

---

## Context.GetCurrentGlobalDateTime

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the current date and time in Coordinated Universal Time (UTC).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td><code>this.result = Context.GetCurrentGlobalDateTime();</code></td>
</tr>
<tr>
<td></td>
<td><strong>Result:</strong> Current date and time in UTC</td>
</tr>
</tbody>
</table>

## Context.GetCurrentIdentityUUID

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the Identity UUID of the user currently logged on.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td><code>this.result = Context.GetCurrentIdentityUUID();</code></td>
</tr>
<tr>
<td></td>
<td><strong>Result:</strong> Identity UUID of the user currently logged on.</td>
</tr>
</tbody>
</table>

## Context.GetCurrentSystemDate

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the current date in the time zone of the system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td><code>this.result = Context.GetCurrentSystemDate();</code></td>
</tr>
<tr>
<td></td>
<td><strong>Result:</strong> Current date in the time zone of the system.</td>
</tr>
</tbody>
</table>

## Context.GetCurrentSystemDateTime

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the current date and time in the time zone of the system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td><code>this.result = Context.GetCurrentSystemDateTime();</code></td>
</tr>
<tr>
<td></td>
<td><strong>Result:</strong> Current date and time in the time zone of the system.</td>
</tr>
</tbody>
</table>

## Context.GetCurrentSystemTime

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the current time in the time zone of the system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td><code>this.result = Context.GetCurrentSystemTime();</code></td>
</tr>
<tr>
<td></td>
<td><strong>Result:</strong> Current time in the time zone of the system.</td>
</tr>
</tbody>
</table>

## Context.GetCurrentUserDate

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the current date in the time zone of the user.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td><code>this.result = Context.GetUserDate();</code></td>
</tr>
<tr>
<td></td>
<td><strong>Result:</strong> Current date in the time zone of the user.</td>
</tr>
</tbody>
</table>
Context.GetCurrentUserDateTime

Description

Returns the current date and time in the time zone of the user.

Example

```javascript
this.result = Context.GetCurrentUserDateTime();
Result: Current date and time in the time zone of the user
```

Context.GetCurrentUserTime

Description

Returns the current time in the time zone of the user.

Example

```javascript
this.result = Context.GetCurrentUserTime();
Result: Current time in the time zone of the user
```

Context.GetCurrentUserLanguage

Description

Returns the language specified for the user.

Example

```javascript
this.result = Context.GetCurrentLanguage();
Result: Language specified for the user.
```

Context.GetSystemURL

Description

Returns the URL of the system.

Example

```javascript
this.result = Context.GetSystemURL();
Result: URL of the system, for example, https://my012345.sapbydesign.com.
```

5.2.5.1.4 Currency Conversion (Reuse Library)

This reuse library contains services that you can use for currency-based amounts. For example, you can convert an amount from one currency into another.

CurrencyConversion.GreaterEquals

Description

Indicates whether or not an amount is greater than or equal to another amount. You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.

Example

```javascript
this.amount1.content = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.GreaterEquals(this.amount2);
Result: False
```

CurrencyConversion.GreaterThan

Description

Indicates whether or not an amount is greater than another amount. You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.
### CurrencyConversion.GreaterThan

**Description**
Indicates whether or not an amount is greater than another amount. You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.

**Example**
```javascript
this.amount1.content = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.GreaterThan(this.amount2);
Result: False
```

### CurrencyConversion.LessEquals

**Description**
Indicates whether or not an amount is less than or is equal to another amount. You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.

**Example**
```javascript
this.amount1.content = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.LessEquals(this.amount2);
Result: True
```

### CurrencyConversion.LessThan

**Description**
Indicates whether or not an amount is less than another amount. You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.

**Example**
```javascript
this.amount1.content = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.LessThan(this.amount2);
Result: True
```

### CurrencyConversion.Equals

**Description**
Indicates whether or not an amount is equal to another amount. You can compare amounts with different currency codes. The result depends on the exchange rate maintained in the system.

**Example**
```javascript
this.amount1.content = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.Equals(this.amount2);
Result: False
```

### CurrencyConversion.Add

**Description**
Adds an amount to another amount. You can add amounts with different currency codes and define a target currency code. The result depends on the exchange rate maintained in the system.

**Example**
```javascript
this.amount1.content = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.Add(this.amount2);
Result: Amount value 33 EUR
```
CurrencyConversion.Subtract

**Description**
Subtracts an amount from another amount. You can add amounts with different currency codes and define a target currency code. The result depends on the exchange rate maintained in the system. Negative results are allowed.

**Example**
```javascript
this.amount1.content = 10;
this.amount1.currencyCode = "EUR";
this.amount2.content = 23;
this.amount2.currencyCode = "EUR";
this.result = this.amount1.Subtract(this.amount2);
Result: Amount value -13 EUR
```

CurrencyConversion.Round

**Description**
Rounds an amount to two decimals.

**Example**
```javascript
this.amount.content = 10.3754;
this.amount.currencyCode = "EUR";
this.result = this.amount.Round();
Result: Amount value 10.38 EUR
```

CurrencyConversion.ConvertCurrency

**Description**
Converts an amount into a different currency based on the exchange rate maintained in the system.

**Example**
```javascript
this.amount.content = 10.3754;
this.amount.currencyCode = "EUR";
this.result = this.amount.ConvertCurrency("USD");
Result: Depends on the exchange rate
```

CurrencyConversion.SpellAmount

**Description**
Spells out an amount. The integer part and the decimal part of a decimal number are spelled out separately and returned in the respective fields. If you do not provide the language, the logon language is used as a default.

**Example**
```javascript
var spell;
this.Amount1.content = 10.14;
this.Amount1.currencyCode = "EUR";
this.Language = "EN";
spell = this.Amount1.SpellAmount (this.Language);
this.Integer = spell.IntegerValue;
this.Decimals = spell.DecimalPart;
Result: Integer value: "TEN", decimal part: “FOURTEEN”
```

5.2.5.1.5 Date (Reuse Library)

This reuse library contains services that you can use to compare dates and to perform simple calculations with dates. Furthermore, it contains other useful date features, for example, services that you can use to determine the start or end date of a month related to a specified date.

Date.GreaterEquals

**Description**
Indicates whether or not a date is later than or equal to another date.
| Example | `this.date1 = Date.ParseFromString("20100215");
              this.date2 = Date.ParseFromString("20100402");
              this.result = this.date1.GreaterEquals(this.date2);
              Result: False` |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date.GreaterThan</strong></td>
<td>Indicates whether or not a date is later than another date.</td>
</tr>
</tbody>
</table>
| **Example** | `this.date1 = Date.ParseFromString("20100215");
              this.date2 = Date.ParseFromString("20100402");
              this.result = this.date1.GreaterThan(this.date2);
              Result: False` |
| **Date.LessEquals** | Indicates whether or not a date is earlier than or equal to another date. |
| **Example** | `this.date1 = Date.ParseFromString("20100215");
              this.date2 = Date.ParseFromString("20100402");
              this.result = this.date1.LessEquals(this.date2);
              Result: True` |
| **Date.LessThan** | Indicates whether or not a date is earlier than another date. |
| **Example** | `this.date1 = Date.ParseFromString("20100215");
              this.date2 = Date.ParseFromString("20100402");
              this.result = this.date1.LessThan(this.date2);
              Result: True` |
| **Date.IsBetween** | Indicates whether or not a date is between two specified date values. |
| **Example** | `this.date1 = Date.ParseFromString("20100215");
              this.date2 = Date.ParseFromString("20100402");
              this.date3 = Date.ParseFromString("20100330");
              this.result = this.date2.IsBetween(this.date1, this.date3);
              Result: True` |
| **Date.GetDay** | Returns the day of a date. |
| **Example** | `this.date = Date.ParseFromString("20100215");
              this.result = this.date.GetDay();
              Result: String value 15` |
| **Date.GetMonth** | Returns the month of a date. |
| **Example** | `this.date = Date.ParseFromString("20100215");
              this.result = this.date.GetMonth();
              Result: String value 02` |
### Date.GetYear

**Description**
Returns the year of a date.

**Example**
```javascript
this.date = Date.ParseFromString("20100215");
this.result = this.date.GetYear();
Result: String value 2010
```

### Date.GetWeek

**Description**
Returns the calendar week of a date.

**Example**
```javascript
this.date = Date.ParseFromString("20100215");
this.result = this.date.GetWeek();
Result: String value 07
```

### Date.AddDuration

**Description**
Adds a duration to a date.

**Example**
```javascript
this.date = Date.ParseFromString("20100215");
this.result = this.date.AddDuration(Duration.ParseFromString("P3D"););
Result: Date value 20100218
```

### Date.SubtractDuration

**Description**
Subtracts a duration from a date.

**Example**
```javascript
this.date = Date.ParseFromString("20100215");
this.result = this.date.SubtractDuration(Duration.ParseFromString("P3D"););
Result: Date value 20100212
```

### Date.Delta

**Description**
Returns the delta duration between two specified date values.

**Example**
```javascript
this.date1 = Date.ParseFromString("20100215");
this.date2 = Date.ParseFromString("20100218");
this.result = this.date1.Delta(this.date2);
Result: Duration value P3D
```

### Date.StartOfMonth

**Description**
Returns the start of the month related to a specified date.

**Example**
```javascript
this.date = Date.ParseFromString("20100215");
this.result = this.date.StartOfMonth();
Result: Date value 20100201
```

### Date.EndOfMonth

**Description**
Returns the end of the month related to a specified date.

**Example**
```javascript
this.date = Date.ParseFromString("20100215");
this.result = this.date.EndOfMonth();
Result: Date value 20100228
```
### Date.StartOfYear

**Description**
Returns the start of the year related to a specified date.

**Example**
```
this.date = Date.ParseFromString("20100215");
this.result = this.date.StartOfYear();
Result: Date with value 20100101
```

### Date.EndOfYear

**Description**
Returns the end of the year related to a specified date.

**Example**
```
this.date = Date.ParseFromString("20100215");
this.result = this.date.EndOfYear();
Result: Date with value 20101231
```

### Date(ConvertToGlobalDateTime)

**Description**
Converts a date into a global date and time value.

**Example**
```
this.date = Date.ParseFromString("20100215");
this.result = this.date.ConvertToGlobalDateTime();
Result: Global date time value 20100215000000
```

### Date.ParseFromString

**Description**
Converts a string into a date. The service accepts both the internal format and the ISO format. This service only supports static calls.

**Example**
```
this.date = Date.ParseFromString("20100215");
Result: Date value 20100215
```

### 5.2.5.1.6 Date and Time (Reuse Library)

This reuse library contains a service that you can use to convert a string into a date and time value.

### DateTime.ParseFromString

**Description**
Converts a string into a date and time value. This service only supports static calls. Use this service only when an SAP business object uses the data type `DateTime` in namespace `http://sap.com/xi/BASIS/Global`. However, do not use the `DateTime` data type in your own business object definitions.

**Example**
```
this.dateTime = DateTime.ParseFromString("2008-03-30T10:31:00Z");
Result: Date and time value 2008-03-30T10:31:00Z
```
5.2.5.1.7 Duration (Reuse Library)

This reuse library contains services that you can use to compare durations and to perform simple calculations with durations.

Durations are represented as W3C xsd:duration according to ISO 8601.

Duration.GreaterEquals

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether or not a duration is greater than or equal to another duration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>this.duration1 = Duration.ParseFromString(&quot;P3Y&quot;); this.duration2 = Duration.ParseFromString(&quot;P5Y&quot;); this.result = this.duration1.GreaterEquals(this.duration2); Result: False</td>
</tr>
</tbody>
</table>

Duration.GreaterThan

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether or not a duration is greater than another duration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>this.duration1 = Duration.ParseFromString(&quot;P3Y&quot;); this.duration2 = Duration.ParseFromString(&quot;P5Y&quot;); this.result = this.duration1.GreaterThan(this.duration2); Result: False</td>
</tr>
</tbody>
</table>

Duration.LessEquals

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether or not a duration is less than or equal to another duration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>this.duration1 = Duration.ParseFromString(&quot;P3Y&quot;); this.duration2 = Duration.ParseFromString(&quot;P5Y&quot;); this.result = this.duration1.LessEquals(this.duration2); Result: True</td>
</tr>
</tbody>
</table>

Duration.LessThan

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether or not a duration is less than another duration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>this.duration1 = Duration.ParseFromString(&quot;P3Y&quot;); this.duration2 = Duration.ParseFromString(&quot;P5Y&quot;); this.result = this.duration1.LessThan(this.duration2); Result: True</td>
</tr>
</tbody>
</table>

Duration.AddDuration

<table>
<thead>
<tr>
<th>Description</th>
<th>Adds a duration to another duration. The result is not normalized. Normalization can be forced by setting the parameter ForceNormalization to True.</th>
</tr>
</thead>
</table>
Example

```csharp
this.duration1 = Duration.ParseFromString("P28D");
this.duration2 = Duration.ParseFromString("P10D");
this.result = this.duration1.AddDuration(this.duration2);
Result: Duration value "P38D"
```

This service supports static calls only.

### Duration.AddDuration

**Description**

Add a duration to another duration.

**Example**

```csharp
this.duration1 = Duration.ParseFromString("P28D");
this.duration2 = Duration.ParseFromString("P10D");
this.result = this.duration1.AddDuration(this.duration2);
Result: Duration value "P38D"
```

### Duration.SubtractDuration

**Description**

Subtracts a duration from another duration. The subtraction may not have an unambiguous result (the reason is, for example, that some months have 30 days, others have 31 days). In this case, the service will not return a result. The calculation can be forced by setting parameter `ForceCalculation` to `true`. Then the service, for example, constantly uses 30 days for the duration of each month.

Negative durations are not allowed. Therefore, the first duration value must be greater than the second duration value.

**Example**

```csharp
this.duration1 = Duration.ParseFromString("P1M8D");
this.duration2 = Duration.ParseFromString("P10D");
this.result = this.duration1.SubtractDuration(this.duration2);
Result: As the result is ambiguous the service does not return a result. Instead it returns a corresponding error message.
```

### Duration.ConvertToDays

**Description**

Converts a duration into days.

**Example**

```csharp
this.duration = Duration.ParseFromString("P1Y14D");
this.result = this.duration.ConvertToDays();
Result: Integer value 379
```

### Duration.ConvertToHours

**Description**

Converts a duration into hours.

**Example**

```csharp
this.duration = Duration.ParseFromString("P1Y14D");
this.result = this.duration.ConvertToHours();
Result: Integer value 9101
```
5.2.5.1.8 Global Date and Time (Reuse Library)

This reuse library contains services that you can use to compare global date and time values and to perform simple calculations with global date and time values. Furthermore, it contains other useful global date and time features, for example, services that you can use to determine the start or end of a global date and time of a month.

For `GlobalDateTime` elements, use the data type `GLOBAL_DateTime` of namespace `http://sap.com/xi/BASIS/Global`.

Some services include the `IsUTC` parameter. This parameter indicates whether global date and time is passed in Coordinated Universal Time (UTC) or to the time zone of the user. This parameter is needed because the user interface does not pass the date and time in UTC but in the time zone of the user.

**GlobalDateTime.GreaterEquals**

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether or not a global date and time value is greater than or equal to another global date and time value.</th>
</tr>
</thead>
</table>
| Example                                          | `this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`  
`this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");`  
`this.result = this.gDateTime1.GreaterEquals(this.gDateTime2);`  
Result: False                                                                 |

**GlobalDateTime.GreaterThan**

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether or not a global date and time value is greater than another global date and time value.</th>
</tr>
</thead>
</table>
| Example                                          | `this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`  
`this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");`  
`this.result = this.gDateTime1.GreaterThan(this.gDateTime2);`  
Result: False                                                                 |

**GlobalDateTime.LessEquals**

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether or not a global date and time value is less than or equal to another global date and time value.</th>
</tr>
</thead>
</table>
| Example                                          | `this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`  
`this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");`  
`this.result = this.gDateTime1.LessEqual(this.gDateTime2);`  
Result: True                                                                 |

**GlobalDateTime.LessThan**

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether or not a global time value is less than another global time value.</th>
</tr>
</thead>
</table>
| Example                                          | `this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`  
`this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");`  
`this.result = this.gDateTime1.LessThan(this.gDateTime2);`  
Result: True                                                                 |

**GlobalDateTime.Equals**

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether or not a global date and time value is equal to another global date and time value.</th>
</tr>
</thead>
</table>
| Example                                          | `this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");`  
`this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");`  
`this.result = this.gDateTime1.Equals(this.gDateTime2);`  
Result: False                                                                 |
Example
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.gDateTime2 = GlobalDateTime.ParseFromString("20100402235401");
this.result = this.gDateTime1.Equals(this.gDateTime2);
Result: False

GlobalDateTime.IsBetween
Description
Indicates whether or not a global date and time value is between two other specified global date and time values.

Example
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.gDateTime2 = GlobalDateTime.ParseFromString("20100330235401");
this.gDateTime3 = GlobalDateTime.ParseFromString("20100402235401");
this.result = this.gDateTime2.IsBetween(this.gDateTime1, this.gDateTime3);
Result: True

GlobalDateTime.GetDate
Description
Returns the date of a global date and time value.

Example
this.gDateTime = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.GetDate();
Result: Date value 20100215

GlobalDateTime.GetTime
Description
Returns the time of a global date and time value.

Example
this.gDateTime = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.GetTime();
Result: Time value 104352

GlobalDateTime.GetDay
Description
Returns the day of a global date and time value.

Example
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.GetDay();
Result: String value 15

GlobalDateTime.GetMonth
Description
Returns the month of a global date and time value.

Example
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.GetMonth();
Result: String value 02

GlobalDateTime.GetYear
Description
Returns the year of a global date and time value.

Example
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime.GetYear();
Result: String value 2010
### GlobalDateTime.GetWeek

**Description**
Returns the calendar week of a global date and time value.

**Example**
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.GetWeek();
Result: String value 07
```

### GlobalDateTime.GetHour

**Description**
Returns the hours of a global date and time value.

**Example**
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.GetHour();
Result: String value 10.
```

### GlobalDateTime.GetMinute

**Description**
Returns the minutes of a global date and time value.

**Example**
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.GetMinute();
Result: String value 43
```

### GlobalDateTime.GetSecond

**Description**
Returns the seconds of a global date and time value.

**Example**
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.GetSecond();
Result: String value 52
```

### GlobalDateTime.AddDuration

**Description**
Adds a duration to a global date and time value.

**Example**
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.AddDuration(Duration.ParseFromString("P3DT1H"));
Result: Global date time value 20100218114352
```

### GlobalDateTime.SubtractDuration

**Description**
Subtracts a duration from a global date and time value.

**Example**
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.SubtractDuration(Duration.ParseFromString("P3DT1H"));
Result: Global date time value 20100212094352
```

### GlobalDateTime.Delta

**Description**
Returns the delta duration between two global date and time values.

**Example**
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.gDateTime2 = GlobalDateTime.ParseFromString("20100218114352");
this.result = this.gDateTime1.Delta(this.gDateTime2);
Result: Duration value P3DT1H
```
GlobalDateTime.StartOfMonth

Description
Returns the start of a month related to a global date and time value.

Example
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.StartOfMonth();
Result: Global date time value 20100201000000
```

GlobalDateTime.EndOfMonth

Description
Returns the end of a month related to a global date and time value.

Example
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.EndOfMonth();
Result: Global date time value 20100228235959
```

GlobalDateTime.StartOfYear

Description
Returns the start of a year related to a global date and time value.

Example
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.StartOfYear();
Result: Global date time value 20100101000000
```

GlobalDateTime.EndOfYear

Description
Returns the end of a year related to a global date and time value.

Example
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.EndOfMonth();
Result: Global date time value 20101231235959
```

GlobalDateTime.ConvertToDate

Description
Converts a global date and time value into a date.

Example
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.ConvertToDate();
Result: Date value 20101231
```

GlobalDateTime.ConvertToDateTime

Description
Converts a global date and time value into a date and time value. Use this service only when an SAP business object uses the `DateTime` data type in the `http://sap.com/xi/BASIS/Global` namespace. Do not use the `DateTime` data type in your own business object definitions.

Example
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.ConvertToDateTime();
Result: Date time value 2010-02-15T10:43:52.0000000Z
```

GlobalDateTime.ConvertToLocalDateTime

Description
Converts a global date and time value into a local date and time value. You can specify target time zone. If no time zone is specified, the service converts the global date and time value into the time zone of the user that is currently logged on.

Example
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.ConvertToLocalDateTime();
Result: "Use this service only when an SAP business object uses the `DateTime` data type in the `http://sap.com/xi/BASIS/Global` namespace. Do not use the `DateTime` data type in your own business object definitions.

Example
```csharp
this.gDateTime1 = GlobalDateTime.ParseFromString("20100215104352");
this.result = this.gDateTime1.ConvertToDateTime();
Result: Date time value 2010-02-15T10:43:52.0000000Z
```

GlobalDateTime.ConvertToLocalDateTime
Example

```csharp
this.gDateTime = GlobalDateTime.ParseFromString("20080330120000");
this.result = this.gDateTime(ConvertToLocalDateTime();
Result: Local date time value in the time zone of the user.
this.result = this.gDateTime(ConvertToLocalDateTime("CET");
Result: Local date time value
```

GlobalDateTime.ConvertToLocalNormalisedDateTime

**Description**
Converts a global date and time value into a local normalized date and time value. You can specify a time zone. If no time zone is specified, the time zone of the user that is currently logged on is used.

**Example**

```csharp
this.gDateTime = GlobalDateTime.ParseFromString("20080330120000");
this.result = this.gDateTime(ConvertToLocalNormalisedDateTime());
Result: Local normalized date time value with the time zone of the user.
this.result = this.gDateTime(ConvertToLocalNormalisedDateTime("CET");
Result: Local normalized date time value
```

GlobalDateTime.ParseFromString

**Description**
Converts a string into a global date and time value. The service accepts both the internal format and the ISO format.

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>this.result = GlobalDateTime.ParseFromString(&quot;20100215104352&quot;);</code></td>
</tr>
<tr>
<td><code>Result: Global date time value 20100215104352</code></td>
</tr>
</tbody>
</table>

5.2.5.1.9 Language Code (Reuse Library)

This reuse library contains provides a service for language codes.

LanguageCode.ParseFromString

**Description**
Converts a string into a language code. Before the conversion the system checks if the string is a valid language code. The service accepts ISO codes only.

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>this.description.languageCode = LanguageCode.ParseFromString(&quot;EN&quot;);</code></td>
</tr>
<tr>
<td><code>Result: Language code with content &quot;E&quot;</code></td>
</tr>
</tbody>
</table>

5.2.5.1.10 Local Date and Time (Reuse Library)

This reuse library contains services that deal with local date and time values.

LocalDateTime.ConvertTimeZone

**Description**
Converts a local date and time value from one time zone into another time zone.
### Example
```csharp
this.localDateTime = LocalDateTime.ParseFromString("20100215104352 CET");
this.result = this.LocalDateTime.ConvertTimeZone("UTC");
```
**Result:** Local date time value 20100215094352 UTC

### Description
**LocalDateTime.ConvertToGlobalDateTime**
Converts a local date and time value into a global date and time value.

**Example**
```csharp
this.localDateTime = LocalDateTime.ParseFromString("20080330120000 CET");
this.result = this.LocalDateTime.ConvertToGlobalDateTime();
```
**Result:** Global date time value 20080330100000

**LocalDateTime.ParseFromString**
Converts a string into a local date and time value. The service accepts both the internal format and the ISO format. This service supports static calls only.

**Example**
```csharp
this.result = LocalDateTime.ParseFromString("20100215104352 CET");
```
**Result:** Local date time value 20100215104352 CET

### 5.2.5.1.11 Local Normalised Date and Time (Reuse Library)

This reuse library contains a service that deals with local normalized date and time values.

**LocalNormalisedDateTime.ConvertToGlobalDateTime**
Converts a local normalized date and time value into a global date and time value.

**Example**
```csharp
import ABSL;
var global2;
var locnorm;
var global3;
global2 = GlobalDateTime.ParseFromString("20080330120000");
locnorm = global2.ConvertToLocalNormalisedDateTime();
global3 = locnorm.ConvertToGlobalDateTime();
```
**Result:** Global date time value 20080330120000

### 5.2.5.1.12 Numeric (Reuse Library)

This reuse library contains services that you can use for integers and decimals.

**Numeric.IsNumeric**
Indicates whether a string contains a numeric value or not. This service supports static calls only.

**Example**
```csharp
this.result = Numeric.IsNumeric("1.5A");
```
**Result:** False
### Numeric.ParseFromString

**Description**
Converts a string into a numeric value. Before the conversion the system checks if the string is a valid numeric value. This service supports static calls only.

**Example**
```
this.result = Numeric.ParseFromString("1.5");
```

**Result:** Numeric value 1.5

### Numeric.RoundToString

**Description**
Converts a numeric value into a string by rounding it to a given number of decimal places.

**Example**
```
this.pi = 3.14159;
this.result = this.pi.RoundToString(2);
```

**Result:** "3.14"

### 5.2.5.1.13 Numeric Character (Reuse Library)

This reuse library contains services for codes and identifiers with implementation type NUMC. For information about the data types and their implementation type NUMC, see the Business Center at [https://www.sme.sap.com](https://www.sme.sap.com) under SAP Business ByDesign Community Wiki Info Exchange SAP Solutions OnDemand Studio Public Solution Model. You can find the information in the Published Data Types List.

#### NumericCharacter.ParseFromString

**Description**
Converts a string into a numeric character. Before the conversion the system checks if the string is a valid numeric character. This service supports static calls only.

**Example**
```
this.numcCode.content = Numeric.ParseFromString("05");
```

**Result:** Numeric character value 05

### 5.2.5.1.14 Quantity Conversion (Reuse Library)

This reuse library contains services that you can use to compare quantities and to perform simple calculations of quantities. The services allows to use different units of measure if they are convertible.

#### QuantityConversion.GreaterEquals

**Description**
Indicates whether or not a quantity is greater than or equal to another quantity.
<table>
<thead>
<tr>
<th>Example</th>
<th>Code</th>
</tr>
</thead>
</table>
|          | this.quantity1.Content = 52;  
|          | this.quantity1.unitCode = "EA";  
|          | this.quantity2.Content = 52;  
|          | this.quantity2.unitCode = "EA";  
|          | this.result = this.quantity1.GreaterEquals(this.quantity2);  
|          | Result: True |

**QuantityConversion.GreaterThan**

**Description**
Indicates whether or not a quantity is greater than another quantity.

**Example**
this.quantity1.Content = 52;  
this.quantity1.unitCode = "EA";  
this.quantity2.Content = 52;  
this.quantity2.unitCode = "EA";  
this.result = this.quantity1.GreaterThan(this.quantity2);  
Result: False

**QuantityConversion.LessEquals**

**Description**
Indicates whether or not a quantity is less than or equal to another quantity.

**Example**
this.quantity1.Content = 52;  
this.quantity1.unitCode = "EA";  
this.quantity2.Content = 52;  
this.quantity2.unitCode = "EA";  
this.result = this.quantity1.LessEquals(this.quantity2);  
Result: True

**QuantityConversion.LessThan**

**Description**
Indicates whether or not a quantity is less than another quantity.

**Example**
this.quantity1.Content = 52;  
this.quantity1.unitCode = "EA";  
this.quantity2.Content = 52;  
this.quantity2.unitCode = "EA";  
this.result = this.quantity1.LessThan(this.quantity2);  
Result: False

**QuantityConversion.Equals**

**Description**
Indicates whether a quantity is equal to another quantity or not.

**Example**
this.quantity1.Content = 52;  
this.quantity1.unitCode = "EA";  
this.quantity2.Content = 52;  
this.quantity2.unitCode = "EA";  
this.result = this.quantity1.Equals(this.quantity2);  
Result: True

**QuantityConversion.Add**

**Description**
Adds two quantities. You can add quantities of different units of measure (UoM). If you do not specify a target UoM, the service returns the result in the UoM of the first quantity.
Example:  
```
this.quantity1.Content = 53;
this.quantity1.unitCode = "EA";
this.quantity2.Content = 52;
this.quantity2.unitCode = "EA";
this.result = this.quantity1.Add(this.quantity2);
```

Result: Quantity value 105 EA

### QuantityConversion.Subtract

**Description**
Subtracts a quantity from another quantity. You can subtract quantities of different units of measure (UoM). If you do not specify a target UoM, the service returns the result in the UoM of the first quantity.

**Example**
```
this.quantity1.Content = 53;
this.quantity1.unitCode = "EA";
this.quantity2.Content = 52;
this.quantity2.unitCode = "EA";
this.result = this.quantity1.Subtract(this.quantity2);
```

Result: Quantity value 1 EA

### QuantityConversion.Divide

**Description**
Divides a quantity by another quantity. The result is a decimal value.

**Example**
```
this.quantity1.Content = 58;
this.quantity1.unitCode = "EA";
this.quantity2.Content = 37;
this.quantity2.unitCode = "EA";
this.result = this.quantity1.Divide(this.quantity2);
```

Result: Decimal value 1.56756756756756

---

**5.2.5.1.15 Time (Reuse Library)**

This reuse library contains the following services that you can use, for example, to compare times.

### Time.GreaterEquals

**Description**
Indicates whether or not a time value is greater than or equal to another time value.

**Example**
```
this.time1 = Time.ParseFromString("120157");
this.time2 = Time.ParseFromString("230000");
this.result = this.time1.GreaterEquals(this.time2);
```

Result: False

### Time.GreaterThan

**Description**
Indicates whether or not a time is greater than another value.

**Example**
```
this.time1 = Time.ParseFromString("120157");
this.time2 = Time.ParseFromString("230000");
this.result = this.time1.GreaterThan(this.time2);
```

Result: False
**Time.LessEquals**

**Description**: Indicates whether or not a time value is less than or equal to another value.

**Example**

```csharp
this.time1 = Time.ParseFromString("120157");
this.time2 = Time.ParseFromString("230000");
this.result = this.time1.LessEquals(this.time2);
Result: True
```

**Time.LessThan**

**Description**: Indicates whether or not a time value is less than another value.

**Example**

```csharp
this.time1 = Time.ParseFromString("120157");
this.time2 = Time.ParseFromString("230000");
this.result = this.time1.LessThan(this.time2);
Result: True
```

**Time.GetHour**

**Description**: Returns the hour of a time value.

**Example**

```csharp
this.time = Time.ParseFromString("120157");
this.result = this.time.GetHour();
Result: String value 12
```

**Time.GetMinute**

**Description**: Returns the minutes of a time value.

**Example**

```csharp
this.time = Time.ParseFromString("120157");
this.result = this.time.GetMinute();
Result: String value 01
```

**Time.GetSecond**

**Description**: Returns the seconds of a time value.

**Example**

```csharp
this.time = Time.ParseFromString("120157");
this.result = this.time.GetSecond();
Result: String value 57
```

**Time.Delta**

**Description**: Returns the duration delta between two specified time values.

**Example**

```csharp
this.time1 = Time.ParseFromString("120157");
this.time2 = Time.ParseFromString("230000");
this.result = this.time1.Delta(this.time2);
Result: Duration value PT10H58M3S
```

**Time.ParseFromString**

**Description**: Converts a string into a time value. The service accepts both the internal format and the ISO format.

- This service supports static calls only.
Example

```csharp
this.result = Duration.ParseFromString("120157");
Result: Time value 12:01:57
```

5.2.5.1.16 UUID (Reuse Library)

This reuse library contains services for Universal Unique Identifiers (UUIDs).

**UUID.ParseFromString**

- **Description**: Converts a string into a UUID.
- **Example**

```csharp
this.result.content = UUID.ParseFromString("00163E01023602DD88AFE2F34FC4C3F3");
Result: UUID 00163E01023602DD88AFE2F34FC4C3F3
```

**UUID.Generate**

- **Description**: Generates a UUID.
- **Example**

```csharp
this.result.content = UUID.Generate();
Result: A UUID that is different each time the service is called.
```

5.2.5.1.17 Web Service Utilities

This reuse library contains a service that you can use to call a REST mashup Web service.

**WebServiceUtilities.ExecuteWebService**

- **Description**: Calls a REST mashup Web service. For more information, see Defining the Business Logic to Call a REST Mashup Web Service [page 396].
- **Example**

```csharp
var result = WebServiceUtilities.ExecuteWebService(serviceID, parameters);
var content = result.ResponseContent;
var returnCode = result.ReturnCode;
```

5.2.5.2 Reuse Libraries for Business Areas

5.2.5.2.1 Reuse Libraries for Business Areas Reference

The reuse libraries for business areas provide additional, very specific services to the scripting language that are relevant for functions in the business areas of the SAP solution, for example, in the Financial Management business area. These services are all based on entities provided by SAP.
To use a reuse library for a specific business area, you need to import the namespace of the relevant business area with the `import` keyword, for example, `import AP.FO.ProductDataMaintenance.Global;`. For more information about the `import` keyword, see [Syntax for Implementation of Business Logic](#) [page 178].

**List of Reuse Libraries for Business Areas**

- **Accounting Coding Block Assignment Utilities** [page 224]
  This reuse library contains a service that is based on the `AccountingCodingBlockDistribution` dependent object provided by SAP.

- **Address Snapshot Utilities** [page 225]
  This reuse library contains a service that is based on the `Address Snapshot` business object provided by SAP.

- **Date Time Utilities** [page 225]
  This reuse library contains a service that is based on advanced functions for date and time processing.

- **Financials Process Utilities** [page 225]
  This reuse library contains a service that is relevant for the Financial Management business area.

- **Financials Set of Books Utilities** [page 226]
  This reuse library contains a service that is based on the `Set of Books` business object provided by SAP.

- **Material Utilities** [page 227]
  This reuse library contains a service that is based on the `Material` business object provided by SAP.

- **Price Component Utilities** [page 228]
  This reuse library contains a service that is based on the `PriceAndTaxCalculation` dependent object provided by SAP.

**See Also**

- Basic Reuse Libraries [page 202]

### 5.2.5.2.2 Accounting Coding Block Assignment Utilities

To use this reuse library, you need to import the `AP.IS.CodingBlock` namespace.

**AccountingCodingBlockAssignmentUtilities.ToMessage**

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
<th>Converts the <code>AccountingCodingBlockAssignment</code> subnode of the <code>AccountingCodingBlockDistribution</code> dependent object into the <code>AccountingCodingBlockAssignment</code> global data type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td></td>
<td><code>FormData.ItemAccountingCodingBlockAssignment = this.ToItemAccountingCodingBlockAssignment.ToMessage();</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>In this example, <code>FormData.ItemAccountingCodingBlockAssignment</code> is the <code>AccountingCodingBlockAssignment</code> global data type. <code>this.ToItemAccountingCodingBlockAssignment</code> refers to the <code>AccountingCodingBlockAssignment</code> subnode of the <code>AccountingCodingBlockDistribution</code> dependent object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Result:</strong> Accounting coding block assignment.</td>
</tr>
</tbody>
</table>
5.2.5.2.3 Address Snapshot Utilities

To use this reuse library, you need to import the `AP.FO.Address.Global` namespace.

**AddressSnapshotUtilities.ToForm**

<table>
<thead>
<tr>
<th>Description</th>
<th>Converts an address into a format that can be used in a print form. The data stored in an instance of the Address Snapshot business object is converted into the FormAddress global data type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td><code>this.SenderCountry = &quot;US&quot;; this.MaximumOfLines = 12; this.UseStreetAddress = true; FormData.PurchaseOrder.SellerPartyAddress = this.SellerParty.AddressSnapshot.ToForm(this.SenderCountry, this.MaximumOfLines, this.UseStreetAddress );</code></td>
</tr>
</tbody>
</table>

In this example, `AddressSnapshot` refers to the Address Snapshot business object. `FormData.PurchaseOrder.SellerPartyAddress` is the FormAddress global data type.

**Result:** Form address, for example, `FormData.PurchaseOrder.SellerPartyAddress.FormattedAddressDescription = "AMS / 166 South High Street / Akron OH 44308 / US"`

5.2.5.2.4 Date Time Utilities

This reuse library contains a service that is based on advanced functions for date and time processing. To use this library, you need to import the `AP.IS.DateAndTime.Global` namespace.

**DateTimeUtilities.GetNextActiveTimePoint**

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the next active time value based on the calendar, direction (+ or -) and the date and time value passed. If you do not provide a calendar, the default calendar is used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td><code>this.CurrentDateTime.Date = Date.ParseFromString(&quot;20080330&quot;); this.CurrentDateTime.Time = Time.ParseFromString(&quot;101010&quot;); this.Direction = &quot;+&quot;; this.CalCode.content = &quot;11&quot;; this.NextDateTime = DateTimeUtilities.GetNextActiveTimePoint(this.CurrentDateTime, this.Direction, this.CalCode);</code></td>
</tr>
</tbody>
</table>

**Result:** Next date value: 31.03.2008, next time value: 000000

5.2.5.2.5 Financials Process Utilities

This reuse library contains a service that is relevant for the Financial Management business area. To use this library, you need to import the `AP.FO.CompanyFinancialsProcessControl.Global` namespace.
### FinancialsProcessUtilities.GetProcessLockStatus

| Description | Returns the process lock status for a given accounting transaction date, company and business transaction type group. Please note that on the user interface of SAP Business ByDesign, process lock is called “Process Control” and accounting transaction date is called “Posting Date” or “Proposed Posting Date”. For information about the process lock, see the SAP Business ByDesign Library under Business Areas ➔ Financial Management ➔ Financial and Management Accounting ➔ General Ledger ➔ Master Data ➔ Companies View ➔ Quick Guide for Companies (General Ledger) ➔ Set the Process Control. |
| Example | this.BTTGroup = "02"; // Customer Invoicing  
this.TransactionDate = Date.ParseFromstring("20110623");  
this/blockingStatus = FinancialsProcessUtilities.ProcessLock( this.TransactionDate, this.BTTGroup, this.ToCompany );  
In this example, this.ToCompany refers to the Company business object provided by SAP.  
Result: Process lock status code, for example, 1 for status “not blocked”. |

### 5.2.5.2.6 Financials Set of Books Utilities

This reuse library contains a service that is based on the Set of Books business object provided by SAP. To use this library, you need to import the AP.FinancialAccounting.Global namespace.

| Description | Returns the start date and the end date of the fiscal year for a given company, set of books, and fiscal year. |
| Example | |
5.2.5.2.7 Material Utilities

This reuse library contains a service that is based on the Material business object provided by SAP. To use this library, you need to import the AP.FO.ProductDataMaintenance.Global namespace.

MaterialUtilities.QuantityConversion

<table>
<thead>
<tr>
<th>Description</th>
<th>Converts a quantity of a material or product into a different unit of measure or quantity specification.</th>
</tr>
</thead>
</table>

**Example**

```java
this.quantity_in.content = 1;
this.quantity_in.unitCode = "XPX";
this.unit_out = "EA";
this.quantity_out = this.toMaterial.QuantityConversion( this.quantity_in, this.unit_out );
```

In this example, toMaterial refers to the Material business object.

**Result:** Quantity value, for example, 20 EA

5.2.5.2.8 Output Management Utilities

This reuse library contains a service that is based on output management functionality. To use this reuse library, you need to import the following namespaces:
OutputManagementUtilities.GetPDF

<table>
<thead>
<tr>
<th>Description</th>
<th>Provides business object data in a portable document format (PDF) document.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>//variables</td>
</tr>
<tr>
<td></td>
<td>var FormTemplateLanguage = &quot;E&quot;;</td>
</tr>
<tr>
<td></td>
<td>var PDF : BinaryObject;</td>
</tr>
<tr>
<td></td>
<td>var FormTemplateCode : OutputRequestFormTemplateCode;</td>
</tr>
<tr>
<td></td>
<td>FormTemplateCode.content = &quot;ZMEI2TXR_PJJE6&quot;;</td>
</tr>
<tr>
<td></td>
<td>//Code is Form Template Header Code</td>
</tr>
<tr>
<td></td>
<td>// Reuse Service Call</td>
</tr>
<tr>
<td></td>
<td>PDF =</td>
</tr>
<tr>
<td></td>
<td>OutputManagementUtilities.GetPDF(this,FormTemplateCode,FormTemplateLanguage)</td>
</tr>
<tr>
<td></td>
<td>;</td>
</tr>
<tr>
<td></td>
<td><strong>Result:</strong> A PDF, which can be used further for storage, for example, as a business object attachment.</td>
</tr>
</tbody>
</table>

5.2.5.2.9 Price Component Utilities

This reuse library contains a service that is based on the PriceAndTaxCalculation dependent object provided by SAP. To use this library, you need to import the AP.FO.PriceAndTax.Global namespace.

PriceComponentUtilities.ToMessage

<table>
<thead>
<tr>
<th>Description</th>
<th>Converts the PriceAndTaxCalculation dependent object into the FormPriceComponent global data type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>FormData.PriceComponent = this.PriceAndTaxCalculation.PriceComponent.ToMessage( );</td>
</tr>
<tr>
<td></td>
<td>In this example, FormData.PriceComponent is the FormPriceComponent global data type.</td>
</tr>
<tr>
<td></td>
<td>this.PriceAndTaxCalculation.PriceComponent refers to the PriceAndTaxCalculation dependent object.</td>
</tr>
<tr>
<td></td>
<td><strong>Result:</strong> Form price component.</td>
</tr>
</tbody>
</table>

5.2.6 Built-In Functions

5.2.6.1 Built-In Functions Reference

The built-in functions of the scripting language extend the scope of the syntax for the implementation of your business logic in a similar way as the basic reuse libraries. However, in contrast to the reuse libraries, you do not need to import a namespace to use the built-in functions.

List of Built-In Functions

The built-in functions are listed in the following:
5.2.6.2  Generic Built-In Functions

These functions bind to more than one data type.

<table>
<thead>
<tr>
<th>Function</th>
<th>Binds to Data Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>All data types except for Boolean data types.</td>
<td>Clears a value.</td>
</tr>
<tr>
<td>IsInitial</td>
<td>All data types except for Boolean data types.</td>
<td>Indicates whether a value is initial or not. Returns a Boolean value.</td>
</tr>
</tbody>
</table>
| ToString | • All base types except for string data types  
• All non-base types that are not structured  
• Amount (http://sap.com/xi/AP/Common/GDT)  
• Quantity (http://sap.com/xi/AP/Common/GDT)  
• LOCAL_DateTime (http://sap.com/xi/BASIS/Global)  
• LOCALNORMALISED_DateTime (http://sap.com/xi/BASIS/Global) | Converts a value into a string. For binary values UTF-8 encoding is used. Invalid byte sequences are replaced by the replacement character (U+FFFD). The replacement character is shown as a black diamond with a white question mark. |

Example:
```javascript
if (!this.numericValue.IsInitial()) {
  this.string = this.numericValue.ToString();
  this.numericValue.Clear();
}
```

5.2.6.3  Built-In Functions for Codes

You can use these functions to read the description of numeric code data types or character-based code data types. If you use a structured data type, the method binds to the structure but not to the content element.

GetDescription

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the description of a code value in the language of the user. For some data types, for example, if you use the RegionCode data type, you also need to provide the ListID element.</th>
</tr>
</thead>
</table>
| Examples    | var codeDescriptionInUserLanguage = code.GetDescription();  
var codeDescriptionFrenchRegionCode = regionCode.GetDescription("FR"); |

GetDescriptionInOtherLanguage

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the description of a code value in a specified language. For some data types, for example, if you use the RegionCode data type, you also need to provide the ListID element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td>var codeDescriptionInEnglish = code.GetDescriptionInOtherLanguage(LanguageCode.ParseFromString(&quot;EN&quot;));</td>
</tr>
</tbody>
</table>
5.2.6.4 Built-In Functions for Strings

These functions contain description and processing functions for character strings.

**Concatenate**

<table>
<thead>
<tr>
<th>Description</th>
<th>Concatenates two strings.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td><code>this.string = &quot;aaabbbccc&quot;; this.result = this.string.Concatenate(&quot;test&quot;);</code></td>
</tr>
<tr>
<td></td>
<td>Result: String value <code>aaabbbccctest</code></td>
</tr>
</tbody>
</table>

**Contains**

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether or not a string contains a substring.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the source string is empty, the result is <code>false</code> and the system raises an information message. If the substring is empty, the result is <code>false</code> and the system raises an error message.</td>
</tr>
<tr>
<td>Example</td>
<td><code>this.string = &quot;aaabbbccc&quot;; this.result = this.string.Contains(&quot;bbb&quot;);</code></td>
</tr>
<tr>
<td></td>
<td>Result: True</td>
</tr>
</tbody>
</table>

**EndsWith**

<table>
<thead>
<tr>
<th>Description</th>
<th>Indicates whether a string ends with a substring or not.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If the source string is empty, the result is <code>false</code> and the system raises an information message. If the substring is empty, the result is <code>false</code> and the system raises an error message.</td>
</tr>
<tr>
<td>Example</td>
<td><code>this.string = &quot;aaabbbccc&quot;; this.result = this.string.EndsWith(&quot;ccc&quot;);</code></td>
</tr>
<tr>
<td></td>
<td>Result: True</td>
</tr>
</tbody>
</table>

**Find**

<table>
<thead>
<tr>
<th>Description</th>
<th>Searches from left to right and returns the position of a substring in a string. Optionally, you can define a search area by specifying the start position only or by specifying the start position as well as the length of the search area. If no substring is found, the function returns the value &quot;-1&quot;. The function returns the value &quot;-1&quot; and raises a message if the system encounters one of the following issues:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The substring is empty (error message)</td>
</tr>
<tr>
<td></td>
<td>The start position is not within the length of the string (error message)</td>
</tr>
<tr>
<td></td>
<td>The length succeeds the length of the string (error message)</td>
</tr>
<tr>
<td></td>
<td>The source string is empty (information message)</td>
</tr>
<tr>
<td>Example</td>
<td><code>this.string = &quot;aaabbbcccbbb&quot;; this.result = this.string.Find(&quot;bbb&quot;);</code></td>
</tr>
<tr>
<td></td>
<td>Result: Integer value 3</td>
</tr>
<tr>
<td></td>
<td>Specifying the start position of the search area:</td>
</tr>
<tr>
<td></td>
<td><code>this.string = &quot;aaabbbcccbbb&quot;; this.result = this.string.Find(&quot;bbb&quot;, 5);</code></td>
</tr>
<tr>
<td></td>
<td>Result: Integer value 9</td>
</tr>
<tr>
<td></td>
<td>Specifying the start position and length of the search area:</td>
</tr>
<tr>
<td></td>
<td><code>this.string = &quot;aaabbbcccbbb&quot;; this.result = this.string.Find(&quot;bbb&quot;, 5, 3);</code></td>
</tr>
<tr>
<td></td>
<td>Result: Integer value -1</td>
</tr>
</tbody>
</table>
FindLast

Description
Searches from right to left and returns the position of a substring in a string. Optionally, you can define a search area by specifying the start position only or by specifying the start position and the length of the search area. If no substring is found, the function returns the value "-1".

The function returns the value "-1" and raises a message if the system encounters one of the following issues:

- The substring is empty (error message)
- The start position is not within the length of the string (error message)
- The length succeeds the length of the string (error message)
- The source string is empty (information message)

Example
```plaintext
this.string = "aaabbbcccbbb";
this.result = this.string.FindLast("bbb");
Result: Integer value 9
```

Specifying the start position of the search area:
```plaintext
this.string = "aaabbbcccbbb";
this.result = this.string.FindLast("bbb", 8);
Result: Integer value 3
```

Specifying start position and length of the search area:
```plaintext
this.string = "aaabbbcccbbb";
this.result = this.string.FindLast("bbb", 8, 5);
Result: Integer value 3
```

FindLastRegEx

Description
Searches from right to left and returns the position of a regular expression pattern in a string. Optionally, you can define a search area by specifying the start position only or by specifying the start position and the length of the search area. If no substring is found, the function returns the value "-1".

The function returns the value "-1" and raises a message if the system encounters one of the following issues:

- The substring is empty (error message)
- The start position is not within the length of the string (error message)
- The length succeeds the length of the string (error message)
- The source string is empty (information message)

For information about the operators that you can use in a regular expression pattern, see the ABAP Keyword Documentation.

Example
```plaintext
this.string = "aaa bbb ccc bbb";
this.result = this.string.FindRegEx("b\s+");
Result: Integer value 12
```

Specifying the start position of the search area:
```plaintext
this.string = "aaa bbb ccc bbb";
this.result = this.string.FindLastRegEx("b\s+", 8);
Result: Integer value 4
```

Specifying start position and length of the search area:
```plaintext
this.string = "aaa bbb ccc bbb";
this.result = this.string.FindLastRegEx("b\s+", 8, 1);
Result: Integer value -1
```

FindRegEx

Description
Searches from left to right and returns the position of a regular expression pattern in a string. Optionally, you can define a search area by specifying the start position only or by specifying the start position and the length of the search area. If no substring is found, the function returns the value "-1".

The function returns the value "-1" and raises a message if the system encounters one of the following issues:

- The substring is empty (error message)
- The start position is not within the length of the string (error message)
- The length succeeds the length of the string (error message)
- The source string is empty (information message)

For information about the operators that you can use in a regular expression pattern, see the ABAP Keyword Documentation.
### Example
```java
this.string = "aaa bbb ccc bbb";
this.result = this.string.FindRegEx("b\s+");
Result: Integer value 4
```

Specifying the start position of the search area:
```java
this.string = "aaa bbb ccc bbb";
this.result = this.string.FindRegEx("b\s+", 6);
```

Specifying start position and length of the search area:
```java
this.string = "aaa bbb ccc bbb";
this.result = this.string.FindRegEx("b\s+", 6, 3); Result: Integer value —1
```

<table>
<thead>
<tr>
<th>Length</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
|        | Returns the length of a string. | ```java
this.string = "test";
this.result = this.string.Length();
Result: Integer value 4
``` |

<table>
<thead>
<tr>
<th>Matches</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
|        | Indicates whether or not the source string matches the regular expression pattern. The function raises a message and returns the result `false`, if the system encounters one of the following issues: - The regular expression pattern is empty (error message) - The regular expression pattern is invalid (error message) - The source string is empty (information message) | ```java
this.string = "entitlement";
this.RegEx = "entitle.ent";
this.result = this.string.Matches(this.RegEx);
Result: True
``` |

<table>
<thead>
<tr>
<th>Replace</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
|        | Replaces all occurrences of a substring in a string with another substring. If the substring is empty, the function returns the source string. If the source string is empty, the function returns an empty string. | ```java
this.string = "aaabbbccc";
this.result = this.string.Replace("bbb", "test");
Result: String value aaatestccc
``` |

<table>
<thead>
<tr>
<th>ReplaceRegEx</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
|              | Returns all locations in a string with a substring that match a regular expression pattern. Optionally, you can define a search area by specifying the start position only or by specifying the start position and the length of the search area. If the substring is empty or the regular expression pattern is invalid, the function returns the source string and raises an error message. If the source string is empty, the function returns an empty string and raises an information message. For information about the operators that you can use in a regular expression pattern, see the ABAP Keyword Documentation. | ```java
this.string = "aaa bbb ccc bbb";
this.result = this.string.ReplaceRegEx("b\s+","test");
Result: String value aaatestccc
``` |
StartsWith

Description: Indicates whether or not a string starts with a substring.
If the source string is empty, the result is false and the system raises an information message. If the substring is empty, the result is false and the system raises an error message.

Example:
```csharp
this.string = "aaabbbccc";
this.result = this.string.StartsWith("ccc");
Result: False
```

Substring

Description: Returns a substring of a string starting at a specified position. Optionally, the length of the substring can be defined.
The function raises a message and returns an empty string, if the system encounters one of the following issues:
- The start position is not within the length of the string (error message)
- The length succeeds the length of the string (error message)
- The source string is empty (information message)

Example:
```csharp
• Substring of a string starting at a specified position:
  this.string = "aaabbbccc";
  this.result = this.string.Substring(3);
  Result: String value bbccc
• Defined length of the substring:
  this.string = "aaabbbccc";
  this.result = this.string.Substring(4,3);
  Result: String value bbc
```

ToLowerCase

Description: Converts a string into lower case.
If the source string is empty, the system returns an empty string and raises an information message.

Example:
```csharp
this.string = "AAAbbbCCC";
this.result = this.string.ToLower();
Result: String value aaabbbccc
```

ToUpperCase

Description: Converts a string into upper case.
If the source string is empty, the system returns an empty string and raises an information message.

Example:
```csharp
this.string = "AAAbbbCCC";
this.result = this.string.ToUpper();
Result: String value AAABBBCCC
```

Trim

Description: Deletes leading and trailing characters. It is not relevant for the result, in which order you pass the characters to the system. If you do not specify a character, the system removes all blank characters.
If the source string is empty, the system raises an information message and returns an empty string.

Example:
```csharp
this.string = " aaabbbccc ";
this.result = this.string.Trim();
Result: String value aaabbbccc
```

TrimLeft

Description: Deletes leading characters. It is not relevant for the result, in which order you pass the characters to the system. If you do not specify a character, the system removes all blank characters.
If the source string is empty, the system returns an empty string and raises an information message.
Example

```csharp
this.string = "xyz xyz abcd"
this.result = this.string.TrimLeft("zyx");
Result: String value xyz abcd
```

**TrimRight**

**Description**
Deletes trailing characters. It is not relevant for the result, in which order you pass the characters to the system. If you do not specify a character, the system removes all blank characters. If the source string is empty, the system returns an empty string and raises an information message.

**Example**

```csharp
this.string = "xyz xyz abcd"
this.result = this.string.TrimRight("bd ca");
Result: String value xyz xyz
```

### 5.2.6.5 Built-In Functions for Numeric Characters

This function binds to code and identifier data types with the **NUMC** implementation type.

You can use the **Repository Explorer** tool window to explore the SAP data types that are released with the public solution model (PSM) and their implementation type. For more information, see **Repository Explorer** [page 53].

**ToNumeric**

**Description**
Converts a numeric character into a numeric value.

**Example**

```csharp
var numeric = numcCode.ToNumeric();
```

### 5.2.6.6 Built-In Functions for Identifiers

You can use these functions to remove leading zeros from or add leading zeros to numeric identifier data types and character-based identifier data types.

As a prerequisite for this function, the alpha-conversion indicator for the data type must be set.

**AddLeadingZeros**

**Description**
Adds leading zeros to the identifier.

**Example**

```csharp
this.productID = this.productID.AddLeadingZeros();
```

**RemoveLeadingZeros**

**Description**
Removes leading zeros from the identifier.

**Example**

```csharp
this.productID = this.productID.RemoveLeadingZeros();
```
5.2.7 Basic Data Types

5.2.7.1 Behavior of Data Types

Overview

The scripting language supports basic data types such as string, numeric and Boolean values. In addition, the existing metadata of SAP’s underlying platform for core data types (CDTs) and global data types (GDTs) is made available. The implementation type of the particular data type is taken into account during the mapping of CDTs and GDTs to the basic types of the scripting language. For information about the data types and their implementation type, see the Business Center at https://www.sme.sap.com under SAP Business ByDesign Community Wiki SAP Business ByDesign Studio Public Solution Model. You can find the relevant information in the Published Data Types List.

Basic Data Types

The basic data types that are supported by the SDK are listed in the following table:

<table>
<thead>
<tr>
<th>Basic Type</th>
<th>Implementation Type</th>
<th>Properties</th>
<th>XSD Data Type</th>
<th>SAP GDT or SAP CDT, For Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td></td>
<td></td>
<td>xsd:boolean</td>
<td>Indicator</td>
</tr>
<tr>
<td>Numeric (FLTP)</td>
<td>INT1</td>
<td>1-byte integer (internal). Value range: 0 to 255. The MinimumValue attribute and the MaximumValue attribute define the business contract of the data type. These attributes do not limit the value range technically.</td>
<td>xsd:unsignedByte</td>
<td>SMALLNONNEGATIVEINTEGER_DecimalValue</td>
</tr>
<tr>
<td>INT4</td>
<td>INT4</td>
<td>4-byte integer (internal). Value range: -2(^147,483,648) to +2(^147,483,647).</td>
<td>xsd:int</td>
<td>NumberValue</td>
</tr>
<tr>
<td>DEC</td>
<td>DEC</td>
<td>Decimal number with fixed length (MaximumDigits) and fixed number of fractions (MaximumFractions).</td>
<td>xsd:decimal</td>
<td>Amount.content</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Quantity.content</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Measure.content</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DecimalValue</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SMALLNONNEGATIVEE_Ratio</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Percent</td>
</tr>
<tr>
<td>Basic Type</td>
<td>Implementation Type</td>
<td>Properties</td>
<td>XSD Data Type</td>
<td>SAP GDT or SAP CDT, For Example:</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>DF34_DEC</td>
<td></td>
<td>Decimal floating point numbers of this type are represented internally with 34 decimal places in accordance with the IEEE-754-2008 standard (DF34_DEC). Value range: numbers between 1E6145(1E-34 - 1) and -1E-6143 for the negative range, 0 and +1E-6143 and 1E6145(1E-34) for the positive range. Values lying between these ranges are in the subnormal range and are rounded. Outside the subnormal range, each 34-digit decimal number can be represented precisely by such a decimal floating point number.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Numeric</td>
<td>NUMC</td>
<td>Character string containing the digits 0 to 9 only. Valid content of a numeric text field. The maxLength property specifies the maximum length of the string.</td>
<td>xsd:token</td>
<td>AccountingPeriodID FiscalYearID</td>
</tr>
<tr>
<td>String</td>
<td>CHAR, SSTR, and STRG</td>
<td>Character string containing any alphanumeric characters, including special characters. The SAP solution is a Unicode system with the system code page UTF-16. A character has a length of two bytes. The maxLength property specifies the maximum length of the string.</td>
<td>xsd:token</td>
<td>Text.content LANGUAGEINDEPENDENT_SHORT_Name LONG_Description CONTENT BusinessTransactionDocumentID ID EmailURI ApprovalStatusCode Amount.currency Code</td>
</tr>
<tr>
<td>Binary</td>
<td>RSTR, RAW</td>
<td>String of bytes</td>
<td>xsd:base 64</td>
<td>BinaryObject in particular: usage of AttachmentFolder dependent object</td>
</tr>
<tr>
<td>UUID</td>
<td>XSDUUID_RAW</td>
<td>Universally unique identifier</td>
<td>xsd:token</td>
<td>UUID</td>
</tr>
<tr>
<td>Date</td>
<td>DATS</td>
<td>Gregorian date “YYYY-MM-DD”</td>
<td>xsd:xsd:Date</td>
<td>Date</td>
</tr>
<tr>
<td>GlobalDate</td>
<td>TIME</td>
<td>Time “HH:MM:SS”</td>
<td>xsd:time</td>
<td>Time</td>
</tr>
<tr>
<td>Time</td>
<td>TIMS</td>
<td>Duration, for example “P1Y1M2DT4H12M40S”</td>
<td>xsd:duration</td>
<td>Duration</td>
</tr>
<tr>
<td>Duration</td>
<td>XSDDURATION_ISO</td>
<td>Language code, for example “E”</td>
<td>xsd:language</td>
<td>LanguageCode</td>
</tr>
</tbody>
</table>

For more information, see the data type definitions of the World Wide Web Consortium (W3C) at [http://www.w3.org/TR/xmlschema-2/](http://www.w3.org/TR/xmlschema-2/).
Validation by the User Interface and Web Service Runtime

The user interface (UI) runtime of the SAP solution validates the user input according to the data type definition. The following data type properties are checked in particular:

- For strings: Maximum length
- For numeric values: Minimum and maximum value
- For Boolean expressions: Usually represented by a checkbox
- For date, time, and duration values: Defined length and content

As a consequence, you do not need to implement validations in your script files to ensure that the user input matches the data type definition. The SAP web service runtime validates the service input according to the data type definition (XSD schema validation).

Internal Variables of the Scripting Language

Internal string variables defined in the scripting language are not restricted in length. “Not restricted” means that the variables are limited by the memory quota of a session, which is defined by the system administrator.

Internal numerical variables defined in the scripting language are of the DF34_DEC implementation type.

```plaintext
var string = this.ShortText; // string is a unrestricted string
var i = this.IntegerValue; // i is of DF34_DEC type
In the following example, a very long text is used:
var string = "Lorem ipsum dolor sit amet, consectetur ..."
// very long text, containing "culpa" at position 1000+ and "augue" at position 2000+
position = string.Find("culpa");
string.Substring(position); // result: "culpa" ... (not restricted)
position = string.Find("augue");
this.ShortNote = string.Substring(position); // result: "augue" duis ... (40 char)
```

Conversion

Syntax

If variables are typed with data types that belong to the same basic type in the scripting language, they are implicitly converted by the scripting language. For example:

```plaintext
this.ID = this.ShortName;
if ( this.Amount.content > this.IntegerValue ) ... 
```

If variables are typed with data types that belong to different basic types in the scripting language, they need to be converted by using conversion functions, for example, `ToString()`, `Numeric.ParseFromString()`, `GlobalDateTime.ConvertToDate()`.
Behavior

If you perform operations on string variables in your script files, you need to ensure that the length of the result strings match the data type definition. For example:

```java
// ShortName is typed with LANGUAGEINDEPENDENT_SHORT_Name -> length = 10
// LongName is typed with LANGUAGEINDEPENDENT_LONG_Name -> length = 40
this.ShortName = this.LongName.Substring(0, 10);
```

The program terminates when saving if the length of the `LongName` field is greater than 10 characters and no check exists.

If the content of a variable is outside the range that is specified by the data type definition, the following can occur as a consequence:

- Program termination, for example, an overflow error
- A system message
- An implicit conversion, for example, a cutoff

Variables of unrestricted length, for example, variables of the `LANGUAGEINDEPENDENT_Text` data type, are truncated at a length of 255 characters as soon as they are saved to the database. For texts that are longer than 255 characters, use the `TextCollection` dependent object.

For variables that are typed with numerical data types, rounding is performed according to the definition of the target data type. While the scripting language uses `decfloat34` internally, the values are rounded according to the definition of the data type as soon as the value is assigned to a business object element.

```java
Note: The `Quantity` content data type is specified as a decimal value with 31 digits and 14 fractions.

```java
this.IntegerValue1 = 1 / 3; // result: 0
this.IntegerValue2 = 2 / 3; // result: 1
var num = 1/3;
this.IntegerValue3 = num * 3; // result: 1
this.IntegerValue4 = this.IntegerValue1 * 3; // result: 0
this.Quantity1.content = 1 / 3000000000; // result: 0.00000000033333
this.Quantity2.content = 2 / 3000000000; // result: 0.00000000066667
num = 1/3000000000;
this.Quantity3.content = num * 3000000000; // result: 1
this.Quantity4.content = this.Quantity1.content * 3000000000; // result: 0.99999
this.Quantity5.content = 1 / 3000000000000000000; // result: 0
num = 1/3000000000000000000;
this.Quantity6.content = num * 3000000000000000000; // result: 1
this.Quantity7.content = this.Quantity5.content * 3000000000000000000; // result: 0
num = this.Quantity5.content;
this.Quantity8.content = num * 3000000000000000000; // result: 0
```

Additional Properties for Identifier Data Types

For identifier data types, additional properties are specified in the GDT definition:
- Upper case conversion
- Alpha conversion
  Alpha conversion fills purely numeric user input like “4711” with leading zeros from the left side to allow better sorting in character fields. For example, if alpha conversion is not used, an alphabetic sorting would lead to sorting results such as: “1”, “10”, “100”, “1000”, “2”, “20”, “200”, and so on.

The UI runtime performs upper case conversion and alpha conversion automatically. The scripting language, however, does not perform these types of conversion automatically. Therefore you need to ensure this in your script files.

### ID data type (supporting upper case conversion and alpha conversion)

```javascript
var string = "123456789a123456789b123456789c123456789d123456789e123456789f123456789g123456789h123456789i123456789j123456789k123456789l123456789m123456789n123456789o123456789p123456789q123456789r123456789s123456789t123456789u123456789v123456789w123456789x123456789y123456789z123456789A123456789B123456789C123456789D123456789E123456789F123456789G123456789H123456789I123456789J123456789K123456789L123456789M123456789N123456789O123456789P123456789Q123456789R123456789S123456789T123456789U123456789V123456789W123456789X123456789Y123456789Z";
this.ID = string;
position = this.ID.Find("A"); // result: -1
```

The result is -1 because the content of the `ID` is not automatically converted to upper case. To convert to upper case, use the `ToUpperCase` built-in function, for example:

```javascript
var string = "123456789a123456789b123456789c123456789d123456789e123456789f123456789g123456789h123456789i123456789j123456789k123456789l123456789m123456789n123456789o123456789p123456789q123456789r123456789s123456789t123456789u123456789v123456789w123456789x123456789y123456789z123456789A123456789B123456789C123456789D123456789E123456789F123456789G123456789H123456789I123456789J123456789K123456789L123456789M123456789N123456789O123456789P123456789Q123456789R123456789S123456789T123456789U123456789V123456789W123456789X123456789Y123456789Z";
this.ID = string.ToUpperCase();
position = this.ID.Find("A"); // result: 9
```

### Variables Typed with Code Data Types

Code data types can be mapped to the string basic type or the numeric character basic type. The behavior of a variable typed with a code data type, for example, `PriorityCode`, `ApprovalStatusCode` and `Amount.currencyCode`, is as follows:

- The variable can contain alphanumeric characters or numeric characters. The value range is then defined by fixed or configurable code lists.
- A variable can be assigned, compared, and so on, to a string data type ("=" , "==", "+") if the variable is typed with a code data type mapped to the string basic type. If the code data type, however, is mapped to the numeric character basic type, you need to use the `ToString` built-in function in order to assign or compare the variable to a string data type.
- The variable cannot be assigned, compared, and so on, to other code data types ("=" , "==", "+")

The UI runtime performs a content check of the code list. The scripting language, however, does not perform this check automatically. Therefore you need to ensure this in your script files.

```javascript
var string = "123456789a123456789b123456789c123456789d";
this.ApprovalStatusCode = string; // result: "12" // Note: "12" is not an allowed code value. // Note: Assignment between IDs is not allowed.
this.ApprovalStatusCode = this.DeliveryPriorityCode;
```
5.3 Front-End Script for the UI Designer

5.3.1 Front-End Script Reference

Overview

The front-end script is the programming language used in the user interface designer (UI designer) to define a set of rules and the logic of the behavior of UI controls. For example, you can write a front-end script to enable an edit button only when a field is selected on the user interface. The front-end script must only be used for user interface related adaptation; it must not contain any business logic.

Front-end scripts are used in the following use cases:

- Dynamic setting of UI properties such as Visible, Enabled, Read-only, Mandatory, and Color
- UI-specific code lists
- Dynamic calls of event handlers (for example, for dynamic OBN links)

Script Execution

All scripts (including calculation rules) are executed in a sequential manner together with all other events by the event queue processor in the UI runtime on the client. Parallel execution of scripts is not possible. For each execution of a script, the scripts environment is recreated. All variables, including global variables, are reset.

Due to this execution strategy:

- A large number of executing script instances (such as those triggered by the recalculation of properties) or slow scripts can cause the client to react slowly. This is because all scripts in the event queue are processed before event handler operations that are triggered by end user interaction.
- Events that are triggered by a script — either directly or implicitly due to data manipulation — are also queued and processed only after the script has terminated.

Syntax

The scripts are defined as part of a UI component model. The UI component models are stored and processed as XML files; the script sources are stored as character data sections within these UI component models. The character set of the scripts is based on UTF-8.

Each script and each block used inside control statements is a sequence of statements. For more information, see Lexical Structure of Front-End Scripts [page 242] and Syntax of Front-End Scripts [page 243].

For more code samples, see Front-End Code Snippets [page 247].

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.
Interacting with UI Runtime

The following figure provides an overview of the UI client runtime environment that the front-end script interacts with.

Client Controller Overview

Interacting with Runtime Data

The interaction with runtime data is accomplished by accessing members of the global variable $data. For more information, see Interacting with Runtime Data [page 252].

Interacting with Runtime Component Controller

Each script is executed as an operation of an event handler and runs within the scope of the UI component controller. The global variable $controller provides information about UI component controller’s state and methods to interact with the component controller. For more information, see Global Variable: $controller [page 255].

Using the Text Pool

Scripts should never contain any literals that are intended to be presented to the end user, because they cannot be translated. Texts for the end user should also never be created by concatenating variables or variables and literals, because the order of the elements may have to change in different languages. Instead, you should use modeled texts from the text pool of the UI component.
5.3.2 Lexical Structure of Front-End Scripts

The lexical structure of the front-end scripts contains identifiers, comments, and literals.

Identifiers

Identifiers consist of:

- Letters: a–z, A–Z
- Digits: 0–9
- Special: underscore ( _ )

Identifiers must start with a letter or an underscore. There is no length restriction for identifiers.

We recommend using reasonable identifiers since parsing times are adversely affected by large identifiers.

Comments

Comments follow Java/C# specification:

- Single line comments start with //. The remainder of the line is considered part of the comment.
- Comments delimited by /* and */. All content between the delimiters is considered part of the comment.

Literals

- **String Literals**
  String expressions begin and end with double or single quote marks, and these string expressions are subject to backslash escape.
  The scripting engine does not support any escape sequences in the literals.

In general, we do not recommend using escape sequences.

- Numeric Literals
  - -1234: Numeric integers must only contain decimal digits (other characters such as underscore ( _ ) are not supported).
  - 123.56: Decimals must consist of two strings of decimal digits separated by a period ( . )

See Also

Syntax of Front-End Scripts  [page 243]
5.3.3 Syntax of Front-End Scripts

The syntax of the front-end script contains variables, statements, and expressions.

Variables

The following variables are supported:

- **Global variables prefixed by a $-character**
  The UI runtime framework automatically adds the following global variables into the script environment:
  - $controller: allows interaction with the UI component controller. For more information, see Global Variable: $controller [page 255].
  - $data: allows access to and modification of the data of the UI component. For more information, see Interacting with Runtime Data [page 252].
  - $textpool: allows the use of modeled texts from the text pool of the UI component.
  - $eventArgs: allows accessing event-specific data.
  - $currentrow: for calculation rules of controls used inside iterator elements (for example, cell renderers in the table control), $currentrow provides access to the list row against which the control is bound.

- **Local variables**

The basic types that can be used for scripting correspond to the types used by the UI controller:

- **boolean**
- **integer**: a 32–bit integer value
- **decimal**: a 128–bit data type. Compared to floating-point types, the decimal type has a greater precision and a smaller range, which makes it suitable for financial and monetary calculations.
- **string**: string literals
- **date**: date and time
- **data object references**: references to elements from the data container of the UI component.

The initial type of a variable is determined the first time the variable is assigned a value. Afterwards, only values of the same type can be assigned.

Constants are not supported.

Statements

Simple statements are as follows:

**Assignment statement**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>assignment :=</td>
<td>tuesday = 'Tuesday'</td>
</tr>
<tr>
<td></td>
<td>number = 10</td>
</tr>
</tbody>
</table>
print statement

Syntax

arguments :=
  expression
  | expression, argument_list

print_statement :=
  print expression
  | print arguments

Example

print "SAP Business ByDesign solution"
print "Value", a

Comment
The print() statement, although supported, does not have any practical value for the scripts used in the runtime framework. It is used for internal debugging purposes only.

Control statements are as follows:

if statement

Syntax

if_statement:
  if_predicate block end
  | if_predicate block else_block
  | if_predicate block elseif_statement

if_predicate:
  if expression then
  | if expression NEWLINE

else_block:
  else block end

elseif_statement:
  elseif_predicate block end
  | elseif_predicate block else_block
  | elseif_predicate block elseif_statement

elseif_predicate:
  elif expression then
  | elif expression NEWLINE
### Example
```
if ( true ) then print 'ok' end
if ( true )
  print 'ok'
end

a=1;
if ( a==2 )
  print 'ko' ;
elif ( a==1 )
  print 'ok' ;
else
  print 'ko' ;
end

a=1;
if ( a==2 )
  print 'ko' ;
elif ( a==1 )
b=3;
if ( b==2 )
  print 'ko' ;
elif ( b==1 )
  print 'ko' ;
else
  print 'ok' ;
end
else
  print 'else reached' ;
  print 'ko' ;
end
```

### unless statement

**Syntax**
```
unless_statement :=
  unless_predicate block end
  | unless_predicate block else_block

unless_predicate :=
  unless expression then
  | unless expression NEWLINE;

else_block :=
  else block end
```

**Example**
```
unless ( false )
  print 'ok'
end
```

### ternary statement

**Syntax**
```
ternary_statement :=
  expression ? statement : statement
```

**Example**
```
a=1; b=3; c=3;
((a > 0) and (b == 3)) ? (a=1) : (c=2)
```

**Comment**
In this example, the expression on the left side evaluates to "true". The assignment statement a=1 is executed.
for statement

| Syntax | for_statement :=  
|        |     for_in block end  
|        |     for_in:  
|        |         for reference in expression NEWLINE  |

| Example | result = ''  
|        |     for i in 2..9  
|        |         result = result + i  
|        |     end  |

| Comment | Only integer ranges in the form 2..5 (last item included) and 2...5 (last item excluded) are supported. All other expressions will raise a runtime error. |

while statement

| Syntax | while_statement:  
|        |     while_predicate block end  
|        |     while_predicate:  
|        |         while expression NEWLINE  |

| Example | i=0  
|        |     while i<5  
|        |         print i  
|        |         i=i+1  
|        |     end  |

Expressions

The front-end script interpreter supports the following expressions:

- literals, decimal numbers, integer numbers, constants `true` and `false`
- parentheses to indicate precedence
- method invocations
- variables
- unary expressions
- boolean expressions
- relational expressions
- equality expression
- `not` expression
- range expression: only integer ranges in the form `2..5` (last item included) and `2...5` (last item excluded) are supported.
- `#`, `.-`, `*`, `/`
All statements are expressions, including methods and variables. The last member pushed onto the stack is considered as the return value of a method, there is no return statement. The return value of a calculation rule in the runtime framework is determined in the same way. To return a value from a method or from the front-end script program, use an expression.

Example

```plaintext
a // will return the value of the variable 'a'
result = a // will also return the value of the variable 'a' because the assignment statement is an expression and puts 'a' onto the stack
```

The variable result has been introduced to facilitate adoption of the runtime framework. It is a common cause of error that after an assignment to result (which is a way of moving a value to the stack), another statement moves another value to the stack and this is then used as the result of the calculation rule.

5.3.4 Front-End Code Snippets

Index of Code Samples

The following code snippets are front-end script examples and are grouped thematically:

Lists in Data Container
- Get Lead Selection Data
- Get Lead Selection in a Hierarchical List
- Change Data for All List Entries
- Evaluate a Clicked Link in a List
- Disable a Button Based on a Field in Selected Row in a List
- Set Lead Selection of a List
- Working with Lead Selection in Lists
- Working with Lead Selection in Hierarchical Lists
- Build Up a List and Fill Its Content

Calculation Rules
- Make a Control Read Only Using Calculation Rule
- Accessing Data List Values from Calculation Rules
- Access Description Values of Codes in Scripts and Calculation Rules

Language Constructs
- And/Or/If Clauses
- Break Out of a Loop

Other
- Call Another EventHandler
- Using Textpool for Translatable Text
- Format Amount Field with Right Decimals Depending on Currency Code
- Using $System.Saved
- Prevent the Context from Becoming Dirty
Lists in Data Container

Get Lead Selection Data

Description
Gets lead selection data from a list and changes an unbound data model field that is assigned to attribute SelectedVariant of a pane container. Script is executed OnLeadSelectionChange of DataList:List in Data Model.

Result: In an object worklist, the displayed preview pane changes when a new line is selected.

Example

```javascript
lead = $data.List.LeadSelectedIndex;
if (lead > -1)
    paymenttype = $data.List.Get(lead).PaymentType;
    if (paymenttype == "18")
        $data.UIState._SelectedVariant = "IncomingCheck"
    else
        $data.UIState._SelectedVariant = "NoData"
end
end
```

Get Lead Selection in a Hierarchical List

Description
Gets lead selection data from a hierarchical list.

Example

```javascript
p_lead = $data.List.LeadSelectedIndex;
if (p_lead > -1)
    c_lead = $data.List.Get(p_lead).List.LeadSelectedIndex;
    if (c_lead > -1)
        $controller.HandleEvent("EventOnChild");
    else
        $controller.HandleEvent("EventOnParent");
end
end
```

Change Data for All List Entries

Description
Calculates or changes fields in a list.

Example

```javascript
for i in 0..($data.List.Count-1)
    $data.List.Get(i).Column11 = "Hello World " + i
end
```

Evaluate a Clicked Link in a List

Description
If a link is clicked in a list, and the lead selection does not change, you need to check the event argument. Note that if you have a simple source side dispatching script (only one variable involved), we recommend using the Eventhandler condition instead of a script.
### Example

```javascript
class Example

```row = $eventArgs.RowIdentifier;
if (row)
    paymenttype = $data.DataList.Get(row).PaymentType;
    if (paymenttype == "18")
        $controller.HandleEvent("FireIncomingCheck")
    elseif (paymenttype == "4")
        $controller.HandleEvent("FireOutgoingCheck")
    else
        $controller.HandleEvent("NotSupported")
end end```

### Disable a Button Based on a Field in Selected Row in a List

**Description**
You need to create a variable of type Boolean and initial value "true" at root level. Associate the same with enabled property of the button. In the OnClick event of the button, associate an EventHandler of type script and use the following script.

**Example**

```javascript
class Example

```leadIndex = $data.DataList.LeadSelectedIndex;
if (leadIndex > -1)
    if (status == "2")
        $data.result = false;
    else
        $data.result = true;
end else
    $data.result = true;
end```

### Set Lead Selection of a List

**Description**
Note that the selection applies only to the data which is available on the client.

**Example**

```javascript
class Example

```lead = 2
$data.DataList.SetLeadSelectedIndex(lead)```

### Working with Lead Selection in Lists

**Description**
Obtains a reference to the selected row in a flat list.

**Example**

```javascript
class Example

```row = $data.MyList.LeadSelectedRow```

### Working with Lead Selection in Hierarchical Lists

**Description**
Obtains a reference to the selected row in a hierarchical list.

**Example**

```javascript
class Example

```row = $data.MyList.LeadTreeSelectedRow```

### Build Up a List and Fill Its Content

**Description**
This function may be useful for custom radio button groups. Note that you will have to set the text via a textpool element.

**Example**

```javascript
class Example

```$data.DataList.Add( )
$data.DataList.Add( )
for i in 0..($data.DataList.Count-1)
    $data.DataList.Get(i).Code = i
    $data.DataList.Get(i).Value = "text" + i
end```
Calculation Rules

Make a Control Read Only using Calculation Rule

| Description | In the property dialog of the control, the ReadOnly attribute is available. If you open the editor and chose the Calculation Rule selection, a script window opens. To insert new lines, use the Shift + Enter key combination. In the example below, you must make sure to add IssuerBankCode as an invalidation trigger to execute the script. Note that the value of the last expression is used as the result of a calculation rule. The actual name of the parameter does not matter. This will lead to unwanted side effects if you use any expressions after determining the return value. To avoid this, make sure to do a final expression of the return value at the end of your script. The recommended name for the parameter is result. |

| Example | \begin{verbatim} if ($data.IssuerBankCode == "4711") result = true; else result = false; end // ... any other assignments can happen here result = result \end{verbatim} |

Accessing DataList Values from Calculation Rules

| Description | If you want to get the current value of a datalist field in a calculation rule, you can use $controller.Resolve and the relative path of the field in the datalist. Note that there is limited SAP List Viewer (ALV) functionality for calculated fields in lists. For example, there is no sorting and grouping. In the example below, to ensure that the value is always recalculated correctly, you also have to add the fields LowerBoundaryFiscalYearID_LRL_DE and LowerBoundaryAccountingPeriodID_LRO_DE as invalidation triggers of your calculation rule. |

| Example | \begin{verbatim} if ( $controller.Resolve("./TestRunIndicator_LRI_DE") ) result = $textpool.Lookup("PTK") //Test else result = $textpool.Lookup("PTL") //Update end \end{verbatim} |

Access Description Values of Codes in Scripts and Calculation Rules

| Description | You can access the code value description by adding .Description. |


Language Constructs

And/Or/If Clauses

| Description | You can put parts of the and/or/if clause in one row. |
**Example**

```plaintext
LeadSelection = $data.Interactor1_KA.LeadSelectedIndex
if ( LeadSelection > -1 )
    SelectedRow = $data.Interactor1_KA.Get(LeadSelection)
    if ( SelectedRow.InspectionTypeCode_EP != 5 and
         SelectedRow.Status_ES == 5 and SelectedRow.KIT_KFQ == "NO" and
         $controller.Utils.IsInitial(SelectedRow.BRCA_EM) )
        result = false
    else
        result = true
    end
end
```

**Break Out of a Loop**

**Description**

Useful for conditional breakouts from the loops, such as selecting a checkbox depending on a condition.

**Example**

```plaintext
LeadSelection = $data.DataList.LeadSelectedIndex;
KeyFigureLeadSelected = $data.DataList.Get(LeadSelection).dataField;
i = 0;
while i < $data.DataList.RowCount
    if ( $data.DataList.Get(i).dataField == KeyFigureLeadSelected )
        break;
    else
        $data.ManageExceptions.Get(LeadSelection).checkbox1 = true;
    end
    i = i + 1;
end
```

**Other**

**Call Another Event Handler**

**Description**

The following shows how to call the event handlers "FireIncomingCheck" and "FireAccountingDocument" from a script.

**Example**

```plaintext
if (paymenttype == "18")
    $controller.HandleEvent("FireIncomingCheck")
else
    $controller.HandleEvent("FireAccountingDocument")
end
```

**Using Textpool for Translatable Text**

**Description**

Translatable text should not be hard coded in scripts. You should define a new entry in TextPool with parameter binding.

**Example**

```plaintext
$data.PreviewTitle = $textpool.Lookup("_tUt43CUQ4E0ZhtAgUA0Em")
```

**Format Amount Field with Right Decimals Depending on Currency Code**

**Description**

To format an amount according to a currency code, you need a new statement.

**Example**

```plaintext
amount = 2.1
currencyCode = "EUR"
result = $controller.Utils.AmountToString(amount, currencyCode)
```
Using $System.Saved

**Description**
You can use $System.Saved in your scripts as follows.

**Example**
```sql
if ($controller.Utils.IsInitial($data.$System.Saved))
result = true
else
result = false
end
```

Prevent the Context from Becoming Dirty

**Description**
Note that DirtyState is an unbound field that needs to be added under /Root in the data model to save the dirty state temporarily. To prevent the context from becoming dirty, for example on executing a business object action (and thus preventing the work protect popup from being shown), you have to do as follows.

**Example**
1. Add a script operation before the business object action is executed which saves the current dirty state:
   ```sql
   $data.DirtyState = $data.$System.IsDirty
   ```
2. Add another script operation after the business object action is executed which resets the dirty state to the previous value:
   ```sql
   $data.$System.IsDirty = $data.DirtyState
   ```

Check If an Object-Based Navigation (OBN) Target Is Assigned to the User

**Description**
You can check if an OBN target configured for a floorplan has a valid UI assigned to the current user.

**Example**
```sql
$controller.CheckNavigationTarget("ServiceRequestRootCreateWithReference")
```

**See Also**

Front-End Script Reference  [page 240]

5.3.5 Interacting with Runtime Data

**Overview**

The interaction with runtime data is accomplished by accessing members of the global variable $data. The global variable $data provides access to the controller data container of the UI component by providing a reference to the Root structure.

**Structure of Runtime Data**

The data container of the UI runtime stores the data of the current UI data in the form of named data elements. Types of data elements are as follows:

- **Data Structures**
  Data structures may contain other data elements (fields, structures, and lists). Each member inside a structure can be uniquely identified by its name.
  The Root of the data container is the entry point to all data available for a UI component instance.

- **Data Fields**
Data fields are used to hold data values. The following base types are supported:
- string
- integer
- float
- date
- boolean
- binary
  Note that it is not possible to work with binary data fields in scripting.
- time
  Note that it is not possible to work with time data fields in scripting.
- datetime
  Note that it is not possible to work with datetime data fields in scripting.

- **Data Lists**
  Lists contain zero or more **List Rows**, which are derived from data structures.

The overall composition structure of the data container is determined at design time. It is not possible to add additional members to a data structure at runtime.

### Referencing Runtime Data

When you use the following rules to work with data container elements, you acquire references to data element instances. The data elements are **mutable**.

#### Accessing Members of Structures

Data structure members can be accessed using the syntax `<data structure reference>.<member name>`. The result is a reference to the member data element or `null` if the member does not exist.

Note that when a structure data element is created by the data container, all members are initialized:
- Data field members are created with their initial values.
- Data structure members are created and their members are initialized.
- List members are created and hold zero rows.

**Examples:**

```javascript
// access the field 'NameField' underneat the 'Root' structure
nameFieldRef = $data.NameField

// access a sub structure
subStructRef = $data.SubStruct

// access a member of the sub structure
subFieldRef1 = subStructRef.SubField

// directly access a member of the sub structure
subFieldRef2 = $data.SubStruct.SubField
```

#### Working with Lists

Lists hold zero or more rows of uniform data structures.

Once you have acquired a reference to a list, you may use the following methods and properties to work with the list:
If your list is bound to the backend, the rows that are available in the client may only represent a snapshot of the actual rows the list holds. De-referencing via index is generally discouraged in this case because all indices apply to the current snapshot of rows available in the client.

<table>
<thead>
<tr>
<th>Method/ Property</th>
<th>Returns</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add()</td>
<td>Reference to new list row instances</td>
<td>Adds a new list row and returns a reference to resulting list instance. This may only be used with unbound lists. For bound lists, explicitly modeled list operations have to be used.</td>
<td>myListRef = $data.MyList myListRowRef = myListRef.Add() myListRowRef.ProductTitle = 'SAP Business ByDesign' myListRowRef.Vendor = 'SAP AG'</td>
</tr>
<tr>
<td>Delete(rowIdentifier(string))</td>
<td>Deletes a row identified by the row identifier rowIdentifier. This may only be used with unbound lists. For bound lists, explicitly modeled list operations have to be used.</td>
<td>rowIdentifier = '12' $data.MyList.Delete(rowIdentifier)</td>
<td></td>
</tr>
<tr>
<td>Delete(index(integer))</td>
<td>Deletes the row with the zero-based index from the list. This may only be used with unbound lists. For bound lists, explicitly modeled list operations have to be used.</td>
<td>$data.MyList.Delete(2)</td>
<td></td>
</tr>
<tr>
<td>Clear()</td>
<td>Clears the list (removes all row instances). This may only be used with unbound lists. For bound lists, explicitly modeled list operations have to be used.</td>
<td>$data.MyList.Clear()</td>
<td></td>
</tr>
<tr>
<td>Get(rowIdentifier(string))</td>
<td>Reference to the row instance</td>
<td>Returns the list row with the specified row identifier or null if no such row exists.</td>
<td>rowRef = $data.MyList.Get('12')</td>
</tr>
<tr>
<td>Get(index(string))</td>
<td>Reference to the row instance</td>
<td>Returns the list row at the zero-based index of the list.</td>
<td>rowRef = $data.MyList.Get(2)</td>
</tr>
<tr>
<td>LeadSelectedIndex</td>
<td>integer</td>
<td>Returns the zero-based index of the lead selected row or -1 if no lead selection exists. Returns 999999999 if the lead selected row instance is not available on the client (for example, due to paging).</td>
<td>index = $data.MyList.LeadSelectedIndex</td>
</tr>
<tr>
<td>LeadSelectionNotOnPage</td>
<td>boolean</td>
<td>Returns true if the lead selected row instance is not available on the client (for example, due to paging).</td>
<td>isNotOnPage = $data.MyList.LeadSelectionNotOnPage</td>
</tr>
<tr>
<td>LeadSelectedRow</td>
<td>Reference to the row instance</td>
<td>Returns the lead selected row instance if a lead selection exists and the lead selected row instance is available on the client, which may not always be the case (for example, due to paging)</td>
<td>rowRef = $data.LeadSelectedRow</td>
</tr>
<tr>
<td>LeadTreeSelectedRow</td>
<td>Reference to the row instance</td>
<td>Returns the lead selected row instance for a hierarchical list along the chain of lead selections in the hierarchical list.</td>
<td>rowRef = $data.LeadTreeSelectedRow</td>
</tr>
<tr>
<td>RowCount</td>
<td>integer</td>
<td>Returns the number of rows of the list. This number may exceed the number of rows which are available on the client.</td>
<td>count = $data.RowCount</td>
</tr>
<tr>
<td>EffectiveRowCount</td>
<td>integer</td>
<td>Returns the number of rows that are currently available on the client for this list. The value may be smaller than EffectiveRowCount for paged lists.</td>
<td>count = $data.EffectiveRowCount</td>
</tr>
</tbody>
</table>

**Working with data fields**

© 2012 SAP AG. All rights reserved. • PUBLIC
The scripting environment is agnostic of the CCTS type system used by the UI layer. Do not use scripting to construct or format any representation of a data field value which is intended to be presented to an end user. Use text pool replacements instead.

Once you have acquired a reference to a data field instance, you may work with it like a normal scripting variable. This means you may assign or retrieve the value.

Additional properties exist for data fields:

<table>
<thead>
<tr>
<th>Property</th>
<th>Returns</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>Code value (string)</td>
<td>Returns the code value for a code-list field. The value of the field itself contains the key of the code (which may differ from the code value for hierarchical code lists). Returns null if the code list item for the code has not yet been read or if the field does not have a code list.</td>
<td>codeValue = $data.CodeField.Code</td>
</tr>
<tr>
<td>Description</td>
<td>Description (string)</td>
<td>Returns the description a code-list field (translated human-readable text). Returns null if the code list item for the code has not yet been read or if the field does not have a code list.</td>
<td>codeValue = $data.CodeField.Description</td>
</tr>
<tr>
<td>IsEnabled</td>
<td>Enabled value (boolean)</td>
<td>Returns whether or not the field is enabled. For fields bound to node attributes, the enabled state is determined by the BO / ECO. Client-only fields are always enabled.</td>
<td>enabledValue = $data.IsEnabled</td>
</tr>
<tr>
<td>Locator</td>
<td>string</td>
<td>Returns the binding expression of the field (including row identifiers if the field is contained in list(s)).</td>
<td>locator = $data.SomeField.Locator</td>
</tr>
<tr>
<td>ModelLocator</td>
<td>string</td>
<td>Returns the binding expression of the field (without row identifiers if the field is contained in list(s)).</td>
<td>locator = $data.SomeField.ModelLocator</td>
</tr>
</tbody>
</table>

5.3.6 Global Variable

5.3.6.1 Global Variable: $controller

Overview

The $controller global variable enables script developers to access state information of the UI component controller and to call controller methods.

Controller State Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorOccurred</td>
<td>boolean</td>
<td>Returns a boolean value that indicates whether an error has occurred.</td>
<td>if($controller.ErrorOccurred) // block executed on error end</td>
</tr>
</tbody>
</table>
Transient ErrorOccurred boolean Returns a boolean value that indicates whether or not a transient error has occurred. Transient errors are typically caused by input validation errors and have to be resolved by the end user before normal processing can resume.

\[
\text{if} \left( \text{controller.Transient ErrorOccurred} \right)
\]

// block which executes on transient error end

GetCurrentLanguage string Returns the current logon language of the running session.

\[
\text{data.currentLanguage} = \text{controller.GetCurrentLanguage}();
\]

Utils <object> Returns an object instance which provides general utilities for working with all data types. For more information, see General Utilities for Working with Data [page 256].

\[
\text{utils} = \text{controller.Utils}
\]

StringUtils <object> Returns an object instance which provides utilities for working with strings. For more information, see Utilities for Working with Strings [page 257].

\[
\text{utils} = \text{controller.StringUtils}
\]

5.3.6.2 General Utilities for Working with Data

**Utils** returns an object instance which provides general utilities for working with all data types.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| IsInitial(data(object)) | boolean | This expression is true if the argument data contains the initial value for its type. | a=0

\[
\text{if} \left( \text{controller.Utils.IsInitial(a)} \right)
\]

// this block will execute end

---

Controller Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| HandleEvent(eventName(string)) | Adds the event with the name eventName to the event processing queue of the controller. The method invocation is not synchronous. | \[
\text{controller.HandleEvent('MyEvent')}
\]

| OpenLink(uri(string)) | Opens the target uri in a new window                                         | \[
\text{controller.OpenLink('http://www.sap.com')}
\]

| Resolve(bindingExpression(string)) | Resolves the data field that is addressed by the specified binding expression. For more information, see Binding Expression [page 258]. | \[
\text{controller.Resolve('/Root/LastName')}
\]

| CheckNavigationTarget(outPortName(string)) | Returns a boolean value that indicates whether or not a navigation is possible via the out-port with the name outPortName. This value is true for all dynamic navigations. | \[
\text{isactive} = \text{controller.CheckNavigationTarget('MyOutPort')}
\]
AmountToString(amount(decimal), currencyCode(string))

<table>
<thead>
<tr>
<th>AmountTostring(amount(decimal), currencyCode(string))</th>
<th>string</th>
<th>Returns the amount value as a string formatted according to the currency code. This also ensures that the correct number of decimal digits as defined by the currency is used.</th>
</tr>
</thead>
</table>
| amount = 2.1  
currencyCode = "EUR"  
result = $controller.Utils.AmountToString(amount, currencyCode) |

5.3.6.3 Utilities for Working with Strings

An object instance with utilities for working with strings is exposed via the global variable $controller.StringUtils.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| BeginsWith(text(string), pattern(string)) | boolean | Returns a boolean value which indicates whether text starts with pattern. | if($controller.StringUtils.BeginsWith('SAP Business ByDesign solution', 'SAP'))  
// this block will execute end |
| EndsWith(text(string), pattern(string)) | boolean | Returns a boolean value which indicates whether text ends with pattern. | if($controller.StringUtils.EndsWith('SAP Business ByDesign solution', 'SAP'))  
// this block will not execute end |
| Contains(text(string), pattern(string)) | boolean | Returns a boolean value which indicates whether text contains pattern. | if($controller.StringUtils.Contains('SAP Business ByDesign solution', 'ByD'))  
// this block will execute end |
| AsciiToChar(ascii(integer)) | string | Returns the ascii character (as a string) which corresponds to the code ascii. | if(s = $controller.StringUtils.Contains(32)  
if(s == ' ')  
// this block will execute end |
| IndexOfString(text(string), pattern(string)) | integer | Returns the zero-based first position of pattern in text or -1, if pattern does not occur in text. | position = $controller.StringUtils.IndexOfString('SAP Business ByDesign solution', 'ByD')  
if(position == 4)  
// this block will execute end |
<table>
<thead>
<tr>
<th>Method</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Length</code></td>
<td><code>text(string)</code></td>
<td>Returns the number of characters in text.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>l = $controller.StringUtils.Length('SAP')</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>if(l == 3) // this block will execute</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>end</code></td>
</tr>
<tr>
<td><code>LeftSubString</code></td>
<td><code>text(string), length(integer)</code></td>
<td>Returns the first length characters of text.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>s = $controller.StringUtils.LeftSubString('SAP byd', 3)</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>if(s == 'SAP') // this block will execute</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>end</code></td>
</tr>
<tr>
<td><code>RightSubString</code></td>
<td><code>text(string), length(integer)</code></td>
<td>Returns the last length characters of text.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>s = $controller.StringUtils.RightSubString('SAP ByD', 3)</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>if(s == 'ByD') // this block will execute</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>end</code></td>
</tr>
<tr>
<td><code>Trim</code></td>
<td><code>text(string)</code></td>
<td>Returns text without leading and trailing blanks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>s = $controller.StringUtils.Trim(' SAP ByD')</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>if(s == 'SAP ByD') // this block will execute</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>end</code></td>
</tr>
<tr>
<td><code>LeftTrim</code></td>
<td><code>text(string)</code></td>
<td>Returns text without leading blanks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>s = $controller.StringUtils.LeftTrim(' SAP ByD ')</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>if(s == 'SAP ByD ') // this block will execute</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>end</code></td>
</tr>
<tr>
<td><code>RightTrim</code></td>
<td><code>text(string)</code></td>
<td>Returns text without trailing blanks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>s = $controller.StringUtils.RightTrim(' SAP ByD ')</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>if(s == 'SAP ByD ') // this block will execute</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>end</code></td>
</tr>
<tr>
<td><code>Replace</code></td>
<td><code>text(string), oldValue(string), newValue(string)</code></td>
<td>Replaces all occurrences of oldValue in text to newValue and returns the result as a new string.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>s = $controller.StringUtils.Replace('SAP ByD', 'ByD', 'ByDesign')</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>if(s == 'SAP ByDesign') // this block will execute</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>end</code></td>
</tr>
</tbody>
</table>

### 5.3.6.4 Binding Expression

Using the `Resolve` controller method, script developers can use binding expressions to resolve the value of a field.

```
binding_expression :=
    /Root binding_particles

binding_particles :=
    binding_particle
    | binding_particle binding_particle

binding_particle :=
    /{data element name}
    | /{list name}{row identifier}
    | /{list name}@{row index}
```
Particles

All binding expressions must start with the /Root particle and must specify at least one additional particle. Each particle addresses a member of its parent by name.

- **Example 1:** Assume that a data field named LastName is defined under the root of the data model, the expression /Root/LastName can be used to address the data field.
- **Example 2:** Assume that a data field named LastName is defined as a member of the structure Address, which is defined under the root of the data model, the expression /Root/Address/LastName can be used to address the data field.

List Rows

List rows can be addressed as follows:

- By providing the row identifier
  **Example:** Assume that a data field named LastName is defined as a member of the list Addresses, which is defined under the root of the data model, and the row identifier of the row to be addressed is 4711, the expression /Root/Addresses[4711]/LastName can be used to address the field.

- By providing the (zero-based) row index
  **Example:** Assume that a data field named LastName is defined as a member of the list Addresses, which is defined under the root of the data model, and the row identifier of the row to be addressed is the fifth row in the list, the expression /Root/Addresses/@4/LastName can be used to address the field.

Using indices is not a reliable way of addressing rows in lists that may be paged. Unless modeled otherwise, all bound lists are paged by default.

- Via the lead selection
  **Example:** Assume that a data field named LastName is defined as a member of the list Addresses, which is defined under the root of the data model, the expression /Root/Addresses/Lead/LastName can be used to address the field of the row of the list Addresses, which has the lead selection.
6 Developer Desktop

6.1 Overview of the Developer Desktop

The development tools provided by the SDK are presented in an integrated development environment (IDE) based on Microsoft Visual Studio. The user interface (UI) of the SDK is referred to as the developer desktop. From the developer desktop you access specific features and tools of the development environment. Within the relevant sections, you can also find information on the tools available in the SAP solution to assist you in creating:

- Business configuration
- Analytics
- Business object extensions
- Mashups
- Print forms
- Service integration

Documentation on the available tools in the SAP Business ByDesign solution to help you in building your solution is only an excerpt of the full documentation. For the full documentation, go to http://help.sap.com/on-demand to view the documentation. You can also view the full documentation in the Help Center of the SAP solution.

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages caused by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.
6.2 General Topics

6.2.1 Log On to the Repository

Overview

Before you start working in the SDK, you must connect to the repository, which contains any solutions that you develop as well as the SAP public solution model (PSM) content.
Prerequisites

- You have opened the SDK.
- You have a user and password for the repository. For more information, see User Setup Quick Guide [page 39].

Procedure

1. In My Solutions, click the Log On button.
2. In the Connect to Repository dialog box, select the System that contains the repository to which you want to connect.
   - In most cases, only one System is available — this is your development repository.
   - To edit the repository system connection, from the Administration menu, select Options and Settings and edit the settings on the Connectivity tab.
3. Enter your User ID and Password; then click OK.
   - There is an information icon to the right of each entry field. When you click one of them, the system displays information about the corresponding entry field. Below the entry field, there is a Change Password link. When you click it, SAP Business ByDesign opens where you can change your password: Enter your credentials and click Change Password. You are then prompted to enter a new password.

Result

After you log on to the repository, any solutions that you have created appear in the My Solutions tool window. You can open a solution to view and work on the solution content in the Solution Explorer. You can also create a new solution. For information about creating solutions, see Create a Customer-Specific Solution [page 152].

If you are developing customer-specific solutions and you are logged on to the repository of your development tenant, only the solutions that you have created for a specific customer ID appear in the My Solutions tool window. To create solutions for a different customer ID, choose Administration > Switch Customer. For more information, see Switch a Customer Assignment [page 143].

6.2.2 Report an Incident

Overview

If you encounter a problem during development, you can report an incident to request help in solving the problem. The SAP solution collects context data, such as system information and relevant business data, and attaches it to the incident. This gives information about the system at the time at which the incident occurred and helps SAP Support to better understand the problem. The context data is only useful if you report the incident from the screen on which the problem occurred, however, since it includes information that is specific to this screen. It is important, therefore, that you always report your solution-related incidents from the screen on which they occur in the SAP solution.
You report a problem that is not related to your UI components in the Partner Development work center in the SAP Business ByDesign solution. For example, you create an incident in this work center if you encounter issues with the studio or the user interface designer.

You cannot create an incident from the studio or the UI designer.

Prerequisites

If you create the incident for SAP Support from the SAP solution, you must be logged on to that system with your development user. Development users have key user rights in the SAP Business ByDesign solution and are assigned to the Incidents view of the Application and User Management work center.

Note that you can also create an incident with a standard business user. However, if that business user is not an administrator in the SAP on-demand solution, the system forwards the incident to the responsible development user with administrator rights, who in turn forwards it to SAP Support.

For more information on reporting incidents in the standard SAP solution, see SAP Solution Partner Using Built-In Support in the Business Center at https://www.sme.sap.com under SAP Business ByDesign > Community > Wiki Info Exchange > SAP Solutions OnDemand Studio > Topics in Detail > Documentation > Using Built-In Support.

Procedure

1. Report an Incident Related to Your UI Components
   a. In your solution, click Help Center > Solve Problem or Report Incident.
   b. Enter the required details.

2. Report Another Type of Incident
   a. In the Partner Development work center, click Help Center > Solve Problem or Report Incident.
   b. Enter the required details.

6.2.3 Translation

6.2.3.1 Translation Quick Guide

The SDK provides functions that enable the translation of business configuration content texts and on-screen texts into one of the languages supported by the SAP solution. The translation functions enable you to adapt a solution created using the SDK for a specific country or region. You can export the text of specific project items to a file in XML Localization Interchange File Format (XLIFF). After the XLIFF file is translated, you can then import the translated file into the SDK and check the result. You can also monitor the translation status of project items.

The following table gives an overview of the project items from which you can extract text for translation and the specific text elements that can be translated.
Texts That Can Be Extracted for Translation

<table>
<thead>
<tr>
<th>Project Item</th>
<th>Text Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>User interface (UI) component (*uicomponent) based on a custom business object (*bo)</td>
<td>• Title of floorplan and group</td>
</tr>
<tr>
<td></td>
<td>• Column and menu header</td>
</tr>
<tr>
<td></td>
<td>• Labels for field, checkbox, and radio button</td>
</tr>
<tr>
<td></td>
<td>• Definition, description, instruction, and information</td>
</tr>
<tr>
<td></td>
<td>• Other on-screen texts such as button text, hyperlink text, list item text, explanation text, general text, tab strip text, quick info text, tree node text, bullet list item text, technical text</td>
</tr>
<tr>
<td></td>
<td>The system does not support the extraction of message texts to an XLIFF file.</td>
</tr>
<tr>
<td>Business object (*bo)</td>
<td>• Label of a business object field</td>
</tr>
<tr>
<td></td>
<td>• Tooltip of a business object field</td>
</tr>
<tr>
<td></td>
<td>The system does not support the extraction of message texts to an XLIFF file.</td>
</tr>
<tr>
<td>Business object extension (*xbo)</td>
<td>• Label of an extension field</td>
</tr>
<tr>
<td></td>
<td>• Tooltip of an extension field</td>
</tr>
<tr>
<td></td>
<td>The system does not support the extraction of message texts to an XLIFF file.</td>
</tr>
<tr>
<td>Business adaptation catalog (BAC) element (business option, business topic)</td>
<td>• Description of a business topic</td>
</tr>
<tr>
<td></td>
<td>• Description of a business option</td>
</tr>
<tr>
<td></td>
<td>• Scoping question/statement of a business option</td>
</tr>
<tr>
<td></td>
<td>• Details: Overview and Relevance texts</td>
</tr>
<tr>
<td>Business configuration (BC) set (*bcc)</td>
<td>• Description of a field value</td>
</tr>
<tr>
<td>Data mashup, HTML mashup, URL mashup (*MC.uimashup)</td>
<td>• On-screen texts</td>
</tr>
</tbody>
</table>

Business and Technical Background

XML Localization Interchange File Format (XLIFF)

XML Localization Interchange File Format (XLIFF) is a standard XML-based file format for localization, specifically designed for exchanging bilingual content between systems. This format enables translators to concentrate on the text to be translated by extracting translatable text and separating it from non-translatable text. XLIFF is beneficial as a single format which can be used for translating different kinds of files. With all the advantages of XML-based processing, this format aims to standardize localization and is supported by a wide variety of translators.

For more information, see XML Localization Interchange File Format (XLIFF) [page 265].

Managing Translation

When you create a solution in the SDK, you can translate business configuration content and other solution content into another language. To translate business configuration content and on-screen texts, you must export the translatable content to an XLIFF file on your local computer. Once the XLIFF file is translated, you can import the translated file into the SDK and check the translation status and the result.

For more information, see Managing Translation [page 268].
Tasks

Export Text for Translation
In the SDK you can extract translatable texts from specific project items in a solution. You export this text to a file in XML Localization Interchange File Format (XLIFF) to your local computer for translation into one of the languages supported by the SAP solution. You specify the target language when you export the text.

For more information, see here [page 269].

Import Translated Text
After you have exported text of the translatable project items in a solution to a file in XML Localization Interchange File Format (XLIFF) and it has been translated into the target language, you can import the translated XLIFF file from your local computer to the solution and test the results of the translation in the target language in the SAP solution.

For more information, see here [page 271].

Check Translation Status
You can check the translation status of a project, business configuration content and a UI component by viewing the total number of translatable text strings and translated text strings.

For more information, see here [page 272].

Translate Message Texts

1. Translate message texts that you defined for a business object or a business object extension in the corresponding .bo or .xbo file by using the following syntax based on a sample message:
   
   message NoValidSite text "Enter a valid site";
   message NoValidSite_DE text "Geben Sie einen gültigen Ort ein";
   message NoValidSite_ZH text "";

2. In your action or event script file, raise the language-dependent message using the var expression by using the following syntax:

   ```java
   var userLanguageCode;
   ...
   userLanguage = Context.GetCurrentUserLanguage().ToString();
   if ( userLanguage == "DE" ) { raise NoValidSite_DE.Create("E"); }
   else if ( userLanguage == "ZH" ) { raise NoValidSite_ZH.Create("E"); }
   else { raise NoValidSite.Create("E"); }
   ```

6.2.3.2 Business and Technical Background

6.2.3.2.1 XML Localization Interchange File Format (XLIFF)

Overview

XML Localization Interchange File Format (XLIFF) is a standard XML-based file format for localization, specifically designed for exchanging bilingual content between systems. This format enables translators to concentrate on the text to be translated by extracting translatable text and separating it from non-translatable text. XLIFF is beneficial
as a single format which can be used for translating different kinds of files. With all the advantages of XML-based processing, this format aims to standardize localization and is supported by a wide variety of translators.

**XLIFF File**

As XLIFF is an XML-based file format, it begins with an XML declaration. After the XML declaration comes the XLIFF document itself, enclosed within the `<xliff>` element. An XLIFF document is composed of one or more sections, each enclosed within a `<file>` element. Each `<file>` element corresponds to a file which has to be translated and contains information about the source language and the target language. It could be a project, a business configuration, a project item, such as a floorplan or a business configuration element, such as a BC set or a BAC element. The file element consists of a `<header>` element and a `<body>` element.

The following is an example of the basic structure of an XLIFF file:

```xml
<?xml version='1.0' encoding='utf-8'?>
<xliff xmlns="urn:oasis:names:tc:xliff:document:1.2" version="1.2">
  <file datatype="plaintext" date="2011-12-02T06:22:58Z" original="self" source-language="en" target-language="de">
    <header>
    </header>
    <body>
      <group>
        <trans-unit>
        </trans-unit>
      </group>
    </body>
  </file>
</xliff>
```

The `<header>` element contains metadata about the file which has to be translated. This information depends on the level at which you want to translate.

The following is an example of a `<header>` element in an XLIFF file exported at UI component level:

```xml
<?xml version='1.0' encoding='utf-8'?>
<xliff xmlns="urn:oasis:names:tc:xliff:document:1.2" version="1.2">
  <file datatype="plaintext" date="2011-12-02T06:22:58Z" original="self" source-language="en" target-language="de">
    <header>
      <note>
      </note>
    </header>
    <body>
      <group>
        <trans-unit>
        </trans-unit>
      </group>
    </body>
  </file>
</xliff>
```
<body>

The <trans-unit> element contains the extracted translatable data from the file in the form of translation units represented by <trans-unit> elements. The translatable data within <trans-unit> elements is organized into source and target paired elements. A <source> element stores the source text, and a <target> element stores the translated text. The <target> elements are not mandatory. These <trans-unit> elements can be grouped recursively in <group> elements. For example, if you translate at the project level, the <header> element contains information about the project and each <group> element contains information about an individual UI component.

The following is an example of a <trans-unit> element translated from English to German:

```
<trans-unit xmlns:sap="urn:x-sap:sls-mlt" datatype="plaintext" id="UICT-/BY003505/USINESSOBJECT1_QA.QA-637f86765fbf498ba2cc6017dd6696f0" maxwidth="255" resname="637f86765fbf498ba2cc6017dd6696f0" restype="x-Text" sap:sc="XTXT ">
  <source>Save</source>
  <target>Speichern</target>
</trans-unit>
```

XLIFF is bilingual as each translation unit offers one <source> element and one <target> element. This makes the overall model simple and easy to handle.

### Extraction and Merge Principle

Translation in XLIFF is based on the concept of extracting the translatable text from the original file, and merging it back in place after translation has been done.

The extraction and merge principle is illustrated in the following diagram:
XLIFF Extraction and Merge Principle

All translatable texts and non-translatable texts are extracted from the original file. The parts that are non-translatable are preserved in a temporary file. The translatable parts are stored separately and translated. After translating the translatable texts, both the parts are merged, resulting in the translated file.

See Also
Translation Quick Guide [page 263]

6.2.3.2.2 Managing Translation

Overview

When you create a solution in the SDK, you can translate business configuration content and other solution content into another language. To translate business configuration content and on-screen texts, you must export the translatable content to an XLIFF file on your local computer. Once the XLIFF file is translated, you can import the translated file into the SDK and check the translation status and the result.

Prerequisites

- You have opened a solution in the SDK that contains content that needs to be translated.
- If you want to translate a solution that is already available on SAP Store, you can publish the new language version with a new version of the solution or you must create a patch solution. For more information, see
  - Create a Patch on a Customer's Tenant [page 137]
  - Create a Patch on a Development Tenant [page 141]
- To test translated solution content, you must be assigned the Business User role and have access rights for the work center views where the content is displayed. For more information, see User Setup Quick Guide [page 39].

Process Flow

1. You extract the translatable text strings of all translatable project items in a solution or a specific project item and export the text to an XLIFF file that you save on your local computer. For more information, see Export Text for Translation [page 269].
2. The XLIFF file is translated into the target language using a translation tool that supports XLIFF or an XML editor.
For more information about XLIFF, see XML Localization Interchange File Format (XLIFF) [page 265].

3. You import the translated XLIFF file into the SDK.
   For more information, see Import Translated Text [page 271].

4. You activate the translated project items.
   If you have translated .uicomponent files, you need to update the metadata in the UI designer. To do this, proceed as follows:
   a. Right-click the translated .uicomponent file and select Open in UI Designer.
   b. Switch to edit mode by selecting Edit Display <-> Edit.
   c. Select Tools Update Metadata.
   d. Click the Save and Activate button.

5. Optional: You check the translation status of the solution content.
   For more information, see Check Translation Status [page 272].

6. Optional: To test translated business configuration content, you deploy the business configuration to ensure that you can view the most recently translated business configuration set values in the SAP solution. For information about business configuration content, see Business Configuration Quick Guide [page 272].

7. You log on to the SAP solution as a business user and make sure that you select the target language into which the solution content was translated. You test the translated text by viewing the relevant screens and making sure that all texts appear and are displayed correctly.

See Also

Translation Quick Guide [page 263]

6.2.3.3 Tasks

6.2.3.3.1 Export Text for Translation

Overview

In the SDK you can extract translatable texts from specific project items in a solution. You export this text to a file in XML Localization Interchange File Format (XLIFF) to your local computer for translation into one of the languages supported by the SAP solution. You specify the target language when you export the text.

Prerequisites

You have opened a solution in the SDK that contains content that needs to be translated.

Procedure

1. Extract the translatable text strings of all translatable project items in a solution or a specific project item by doing the following:
   a. In the Solution Explorer, right-click one of the following items:
Project node (to extract the translatable text strings of all project items that support translation)

Floorplan (*.uicomponent)

Business Configuration node (to extract BAC element texts)

Business configuration set (*.bcc)

Business object extension (*.xbo)

b. Select Export Text for Translation.

The Export Text for Translation dialog box opens.

2. In the Export Text for Translation dialog box, select the target language to which you want to translate the text.

3. Enter a name for the XLIFF file.

You can use the default file name or provide a new name. The default file name is in the format Technical Name_Target Language Code.xlf where Technical Name is the name of the project or project item that you selected. Target Language Code is an abbreviation of the target language you selected. This code will be added to the generated XLIFF file.

Examples of Default File Names for Exported XLIFF Files

<table>
<thead>
<tr>
<th>Translation Level</th>
<th>Item</th>
<th>Technical Name</th>
<th>Target Language</th>
<th>XLIFF File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>Project</td>
<td>BY003505</td>
<td>German</td>
<td>BY003505_DE.xlf</td>
</tr>
<tr>
<td>UI component</td>
<td>Object Instance Floorplan</td>
<td>CarParkScreen1_OIF.OIF.uicomponent</td>
<td>Italian</td>
<td>CarParkScreen1_OIF_IT.xlf</td>
</tr>
<tr>
<td>UI component</td>
<td>Object Work List</td>
<td>CarParkScreen1_OWLOWL.uicomponent</td>
<td>Portuguese</td>
<td>CarParkScreen1_OWLOWL_PT.xlf</td>
</tr>
<tr>
<td>UI component</td>
<td>Work Center View</td>
<td>CarParkScreen1_WCVIEW.WCVIEW.uicomponent</td>
<td>Spanish</td>
<td>CarParkScreen1_WCVIEW_ES.xlf</td>
</tr>
<tr>
<td>Business Configuration node</td>
<td>Business adaptation catalog (BAC) element (business option, business topic)</td>
<td>-</td>
<td>Chinese</td>
<td>Business Configuration_ZH.xlf</td>
</tr>
<tr>
<td>Business configuration set</td>
<td>Business Configuration Set</td>
<td>CODE.bcc</td>
<td>French</td>
<td>CODE.bcc_FR.xlf</td>
</tr>
</tbody>
</table>

4. Select the location on your computer where you want to save the exported XLIFF file.

You can save the file in the default location or specify a new location.

5. Click Export.

The file is saved in the specified folder on your local computer.

Result

You have exported the solution content to an XLIFF file on your local computer. You can now translate the XLIFF file into the target language using a translation tool that supports XLIFF or an XML editor. For more information about XLIFF, see XML Localization Interchange File Format (XLIFF) [page 265]. You can also open the XLIFF file in the
studio: from the File menu, select Open File and navigate to the location of the XLIFF file you exported. The file opens in a document window in the code editor.

See Also
Translation Quick Guide [page 263]

6.2.3.3.2 Import Translated Text

Overview
After you have exported text of the translatable project items in a solution to a file in XML Localization Interchange File Format (XLIFF) and it has been translated into the target language, you can import the translated XLIFF file from your local computer to the solution and test the results of the translation in the target language in the SAP solution.

Prerequisites
- You have exported an XLIFF file and saved it on your local computer. For more information, see Export Text for Translation [page 269].
- You have translated the XLIFF file into the target language. For more information about XLIFF, see XML Localization Interchange File Format [page 265].

Procedure
1. In the Solution Explorer, right-click the project, floorplan, business configuration node, or business configuration set for which you want to import translated text and select Import Translation. The Import Translation dialog box opens.
2. Select the location on your computer where you have saved the translated XLIFF file.
3. Click Import.
4. In the Solution Explorer, activate the translated project items by right-clicking the project or the project items and selecting Activate.
5. Optional: If you have imported translated business configuration content, deploy the business configuration by right-clicking the Business Configuration node and selecting Deploy Business Configuration. You must deploy business configuration content to ensure that you can view the most recently translated business configuration set values.
6. Log on to the SAP solution as a business user and make sure that you select the target language into which the solution content was translated.
7. Test the translated text in the SAP solution by viewing the relevant screens and checking that all texts appear and are displayed correctly.

See Also
Translation Quick Guide [page 263]
6.2.3.3.3 Check Translation Status

Overview
You can check the translation status of a project, business configuration content and a UI component by viewing the total number of translatable text strings and translated text strings.

Prerequisites
You have opened the solution in the SDK for which you want to check the translation status.

Procedure
1. In the Solution Explorer, right-click the project, floorplan, business configuration node, or other project item for which you want to check the translation status and select Check Translation Status. The Check Translation Status dialog box opens.
2. Review the total number of translatable text strings and translated text strings, as well as the language to which the text strings have been translated.
   If you check the translation status at project level, the Check Translation Status dialog box displays the translation status of all project items in the solution for which translation is supported.
3. Do one of the following:
   - To export any texts that still need to be translated into one or more languages, select a language and click Export to XLIFF. The Export Text for Translation dialog box opens. For more information, see Export Text for Translation [page 269].
   - If all texts have been translated, click OK. The Check Translation Status dialog box closes.

   If you check the translation status at project level, you can only export text at the project level and not for individual project items.

See Also
Translation Quick Guide  [page 263]

6.3 Business Configuration

6.3.1 Business Configuration Quick Guide

In the SAP Business ByDesign solution, business configuration enables prospective or existing customers to evaluate and set up their system in the production environment to meet company-specific requirements. It also allows customers to adapt and optimize their solutions at any time as business needs change.
You must create business configuration content for all solution capabilities that you create using the studio. You create business configuration content in the studio and anchor your solution in the business adaptation catalog (BAC) so that it is available for activation by your customers.

**Business and Technical Background**

**Business Adaptation Catalog (BAC)**

The full set of capabilities of the SAP solution are outlined in a central business adaptation catalog (BAC). This catalog organizes and structures the capabilities into a hierarchy of business areas, packages, topics, and options. Solutions created in the studio require business configuration content that then appears as elements (BAC elements) in the catalog and allows key users to implement solutions in the production environment.

For more information, see [here](#page-277).

**Business Configuration Content**

You can define the following business configuration content in the studio:

<table>
<thead>
<tr>
<th>Type</th>
<th>Use</th>
<th>How To...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business option (mandatory)</td>
<td>Create a business option when no selective adaptation of business configuration content is required.</td>
<td>...Create a Business Option [page 281]</td>
</tr>
<tr>
<td>Business topic and business options</td>
<td>Create a business topic when you need to provide multiple business options to key users.</td>
<td>...Create a Business Topic and Business Options [page 283]</td>
</tr>
<tr>
<td>Country</td>
<td>Define a country if your solution requires country-specific scoping for a country that is not supported by the standard SAP solution.</td>
<td>...Create a Country and Business Options [page 286]</td>
</tr>
<tr>
<td>Business configuration set (BC set)</td>
<td>Create a BC set to configure the behavior of business processes in your solution and define code list data types that you can use to model fields in a business object. Create a BC set using an SAP business configuration object (BCO) to configure the behavior of standard business processes in the SAP solution.</td>
<td>...Create a BC Set with an Implicit BCO [page 289] ...Create a BC Set Using an SAP BCO [page 290] ...Create a BC Set Using a Custom BCO [page 293]</td>
</tr>
<tr>
<td>Business configuration object (BCO)</td>
<td>Create a custom BCO to define solution-specific business configuration content on which you can base a BC set.</td>
<td>...Create a Business Configuration Object [page 292]</td>
</tr>
<tr>
<td>Business configuration view (BC view)</td>
<td>Create a BC view to allow key users to change the values defined in a BC set during fine-tuning.</td>
<td>...Create a Business Configuration View [page 295]</td>
</tr>
</tbody>
</table>

**Defining Business Configuration Content**

When you create a solution in the studio, you must create business configuration content that contains at least one business option to allow customers who buy your solution to activate the solution in the production environment.

For more information, see [here](#page-278).

**Maintenance of Business Configuration Content**

A solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is **Assembled**. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer’s production tenant.

For information about creating a patch, see [Patches for Customer-Specific Solutions](#page-153).
For information about the change and delete restrictions for business configuration content in a solution that is in maintenance mode, see Maintenance of Business Configuration Content [page 280].

Tools in the SAP Business ByDesign Solution for Business Configuration

To learn about how customers configure their solutions and work with business configuration content, see Configuring Your SAP Business ByDesign Solution.

Tasks

Create BAC Elements

You perform the following tasks using the Business Configuration Wizard:

- **Create a Business Option**
  You must create at least one business option for your solution so that customers can activate the solution in the production environment. You anchor the business option in the business adaptation catalog (BAC) so that it is available in the SAP solution during scoping.
  For more information, see here [page 281].

- **Create a Business Topic and Business Options**
  You can create a business topic to group business options and provide multiple business options for customers to customize your solution.
  For more information, see here [page 283].

- **Create a Country and Business Options**
  You can define a country and create one or more business options if your solution requires country-specific scoping for a country that is not supported by the standard SAP solution.
  For more information, see here [page 286].

- **Change a Business Option or a Business Topic**
  1. In the studio, open the solution that contains the business option or business topic you want to change.
  2. In the Solution Explorer, right-click the Business Configuration node and select Open. The Business Configuration Wizard opens.
  3. Make your changes and then click Save.
  
  To change a business option that you added to a business topic, in the Create Business Options and Assign Solution Content step, use the Change Business Option function.

- **Delete a Business Option or a Business Topic**
  1. In the studio, open the solution that contains the business option or business topic you want to change.
  2. In the Solution Explorer, right-click the Business Configuration node and do one of the following:
     - If you have created only one business option, select Delete.
     - To delete a business topic and all business options you added to the business topic, select Delete.
     - To delete a business option that you added to a business topic without deleting the business topic:
       1. Select Open.
       
       The Business Configuration Wizard opens.
       2. In the Create Business Options and Assign Solution Content step, select the business option you want to delete and then click Delete Business Option.
For information about the change and delete restrictions for business configuration content in a solution that is in maintenance mode, see Maintenance of Business Configuration Content [page 280].

Create Business Configuration Set

For customer-specific solutions:

- You can only create a BC set using an SAP BCO if you are developing the solution on a development tenant. For information, see Lifecycle Management of Customer-Specific Solutions on a Development Tenant [page 139].
- You cannot create BC sets using SAP BCOs in a solution template.

For more information, see here [page 290].

Create Business Configuration Set Wizard

Solution Explorer Business Configuration Create Business Configuration Set

You perform the following tasks using the Business Configuration Set Wizard:

- Create a BC Set with an Implicit BCO
  You can create a business configuration set (BC set) with an implicit business configuration object (BCO) to configure the behavior of business processes in your solution and define code list data types that you can use to model fields in a business object.
  For more information, see here [page 289].

- Create a BC Set Using an SAP BCO
  You can create a business configuration set (BC set) using an SAP business configuration object (BCO) to configure the behavior of standard business processes in the SAP solution. If you create an implementation of an enhancement option which is for single use only, you can also define a BC set using an SAP BCO so that you can enhance the filter values.

- Create a BC Set Using a Custom BCO
  You can create a business configuration set (BC set) using a business configuration object (BCO) that you have defined in your solution.
  For more information, see here [page 293].

- Change a Business Configuration Set
  1. In the studio, open the solution that contains the business configuration set (BC set) you want to change.
  2. If the BC set is based on a business configuration object (BCO) defined in the solution and you want to add or change a field, change the referenced BCO by doing the following:
     a. In the Solution Explorer, clean all BC sets that are based on the BCO by right-clicking the .bcc files and selecting Clean.
     b. Double-click the .bco file.
        The BCO opens in a document window.
     c. On the Fields tab, add or change one or more fields and then save your changes.
     d. In the Solution Explorer, right-click the .bco file and select Activate.
        You can only activate a BC set that is based on a BCO that you changed by opening the BC set in a document window and saving it.
  3. In the Solution Explorer, double-click the .bcc file.
     The Business Configuration Set Wizard opens.
  4. Make your changes, for example, by adding or changing the field values of any fields you add or changed in the BCO.
     You cannot change the name of a BC set.
5. In the Solution Explorer, right-click the .bcc file and select Activate.

For information about the change and delete restrictions for business configuration content in a solution that is in maintenance mode, see Maintenance of Business Configuration Content [page 280].

Create Business Configuration Object

You perform the following tasks using the Business Configuration Object Wizard:

- **Create a Business Configuration Object**
  - **Solution Explorer ➤ Business Configuration ➤ Create Business Configuration Object**
  You can create a business configuration object (BCO) and use it as the basis of a business configuration set (BC set).
  For more information, see here [page 292].

- **Change a Business Configuration Object**
  1. In the Solution Explorer, clean all BC sets that are based on this BCO by right-clicking the .bcc files and selecting Clean.
  2. In the Solution Explorer, double-click the .bco file.
     The business configuration object opens in a document window.
  3. On the Fields tab, add or change one or more fields and then save your changes.
  4. In the Solution Explorer, right-click the .bco file and select Activate.

- **Delete a Business Configuration Object**
  In the Solution Explorer, right-click the .bco file and select Delete.

  To delete a business configuration object that is used in a business configuration set, you must first delete the BC set.

  For information about the change and delete restrictions for business configuration content in a solution that is in maintenance mode, see Maintenance of Business Configuration Content [page 280].

Create Business Configuration View

You perform the following task in the studio and in the UI designer:

- **Create a Business Configuration View**
  - **Solution Explorer ➤ Business Configuration ➤ Create Business Configuration View**
  For more information, see here [page 295].

Check Content in the Business Adaptation Catalog

To check content in the business adaptation catalog (BAC), for example, for testing purposes, you must be assigned the Business User role and have access rights for the Business Configuration work center. For more information, see User Setup Quick Guide [page 39].

1. In the SAP solution, in the Business Configuration work center, open the Implementation Projects view.
2. Select a project and click Edit Project Scope.
   The Edit Project Scope guided activity is displayed. The business topic or business option you created using the Business Configuration Wizard is displayed in the Scoping phase under Scoping Element. The description and scoping questions you defined in the Business Configuration Wizard are displayed in the Questions phase.
Deploy Business Configuration

You deploy your business configuration content to be able to use it in a solution. The system makes the parameter values available in the development environment for testing.

The system assigns all business configuration content that you create to a default business option to simplify testing of business configuration content in a solution. This business option is always activated; therefore, you cannot test partial activation of business configuration content in the development environment. The deployment of business configuration content is independent of any business options you define for your solution.

6.3.2 Business and Technical Background

6.3.2.1 Business Adaptation Catalog

Overview

The full set of capabilities of the SAP solution are outlined in a central business adaptation catalog (BAC). This catalog organizes and structures the capabilities into a hierarchy of business areas, packages, topics, and options. Solutions created in the studio require business configuration content that then appears as elements (BAC elements) in the catalog and allows key users to implement solutions in the production environment.

The customer key users work with the SAP solution in the production environment after it has gone live. This environment contains all elements required to run the customer’s business processes using the SAP solution, and customers can adapt the solution in the Business Configuration work center.

Features

When you create a solution in the studio, you must create business configuration content that contains at least one business option to allow customers who buy your solution to activate the solution in the production environment. A business option represents the most detailed decision level involved in defining or adapting a solution capability. Business options can be hidden or can appear in scoping, fine-tuning, and/or the solution proposal. Each business option specifies a particular way to perform a function. In the standard SAP Business ByDesign system, examples are General Ledger Accounting and Customers. Each business option belongs to a business topic. For information about how to create a business option, see Create a Business Option [page 281].

When you use the Deploy Business Configuration function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the Deploy Business Configuration function deploys all business configuration content independently of any business options you have created.

Optionally, you can also create the following business configuration content:

- Business topic
A business topic relates to specific functions within an application area. In the standard SAP Business ByDesign system, examples are Account Management and Time Administration. Each business topic belongs to a business package and contains a number of business options. For information about how to create a business topic, see Create a Business Topic [page 283].

- Business configuration set (BC set)
  A BC set is a technical container for the configuration settings and corresponding values that are associated with a business option. The content contained within BC sets is deployed to the runtime environment where it is loaded into the underlying configuration tables. Each configuration setting within a BC set has attributes which specify whether it is hidden, visible, or changeable for the customer. For information about how to create a BC set, see Create a Business Configuration Set [page 289]. For information about how to create a BC set using an SAP business configuration object (BCO), see Create a BC Set Using an SAP BCO [page 290].

- Business configuration view (BC view)
  To allow key users to change the values of a BC set during fine-tuning, you can create a BC view and assign this view to the Business Configuration work center.
  For information about how to create a BC view, see Create a Business Configuration View [page 295].

See Also

Business Configuration Quick Guide [page 272]

6.3.2.2 Defining Business Configuration Content

Overview

When you create a solution in the studio, you must create business configuration content that contains at least one business option to allow customers who buy your solution to activate the solution in the production environment. You can assign your business option to a standard business topic or you can create a business topic to which you add your business options.

Prerequisites

- You have created and opened a solution in the SDK.
- You understand the dependencies that exist between the elements in the business adaptation catalog and the constraints that are applied by the selections that key users make during scoping. For more information, see Scoping.
- To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the Business Configuration work center. For more information, see User Setup Quick Guide [page 39].

Process Flow

1. In the studio, you create one or more business options. If you create more than one business option, you group your business options in a business topic. For information about how to create one business option, see Create a Business Option [page 281]. For information about how to create more than one business option, see Create a Business Topic and Business Options [page 283].
2. You define your business options.
   a. You provide a scoping question and details about each business option. During scoping key users select or deselect a business option by answering its associated scoping question. Scoping questions are located in the business adaptation catalog (BAC) at the business topic level. A scoping question is displayed for each business option in the BAC. In the SAP solution, customers can review the details you provide about BAC elements on the Overview and Relevance tabs.
   b. If you want to assign a business configuration set (BC set) to the business option, you specify whether you want customers to be able to adapt the field values of the BC set during fine-tuning. When a key user selects a business option in the scoping phase that is relevant for fine-tuning, the key user can adjust the values of the business configuration sets (BC sets) assigned to this business option by performing a fine-tuning activity.
   For more information about implementing a project in an SAP Business ByDesign production system and the Fine-Tune phase, see Implementing a Project — First Implementation.
   c. You select additional functionality required by your solution by selecting the relevant business topics from the business adaptation catalog list.
   d. You define where you want the business configuration content to be available in the business adaptation catalog.
   For more information about the business adaptation catalog, see Business Adaptation Catalog [page 277] and Configuring Your SAP Business ByDesign Solution.

3. Optional: You create a BC set and assign it to a business option.
   You specify whether key users are able to add or delete the values of specific fields in fine-tuning.
   For more information, see
   • Create a BC Set with an Implicit BCO [page 289]
   • Create a BC Set Using an SAP BCO [page 290]
   • Create a BC Set Using a Custom BCO [page 293]

4. Optional: You create a business configuration view (BC view) for a BC set using the UI designer and assign this view to the Business Configuration work center.
   For more information, see Create a Business Configuration View [page 295].
   You can only create a BC view for a BC set that uses a custom BCO. You cannot create a BC view for an SAP BCO.

5. You activate your business configuration content.
   The system generates the required business configuration object(s) and data type(s).

6. You trigger deployment of your business configuration content to be able to use the content in your solution.
   When you use the Deploy Business Configuration function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the Deploy Business Configuration function deploys all business configuration content independently of any business options you have created.

7. You check that your business configuration content appears in the BAC in the SAP solution.
   For more information, see Business Configuration Quick Guide [page 272].
6.3.2.3 Maintenance of Business Configuration Content

Overview

A solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is *Assembled*. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer’s production tenant.

Change and Delete Restrictions

The following tables give an overview of the change and delete restrictions for business configuration content in a solution that is in maintenance mode.

### Maintenance of Business Topics

<table>
<thead>
<tr>
<th>Action</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change <em>Type of Business Configuration</em></td>
<td>No</td>
</tr>
<tr>
<td>Change <em>Name</em></td>
<td>No</td>
</tr>
<tr>
<td>Change <em>Description</em></td>
<td>Yes</td>
</tr>
<tr>
<td>Change <em>Anchor</em></td>
<td>No</td>
</tr>
<tr>
<td>Change <em>Required Scope</em></td>
<td>With restrictions: You can remove the required scope, but you cannot add required scope.</td>
</tr>
<tr>
<td>Change assigned business options</td>
<td>With restrictions: See <em>Maintenance of Business Options</em></td>
</tr>
<tr>
<td>Delete assigned business options</td>
<td>No</td>
</tr>
<tr>
<td>Change content assigned to business options</td>
<td>No</td>
</tr>
<tr>
<td>Create (and delete) a new business option</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Maintenance of Business Options

<table>
<thead>
<tr>
<th>Action</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change <em>Name</em></td>
<td>No</td>
</tr>
<tr>
<td>Change <em>Description</em></td>
<td>Yes</td>
</tr>
</tbody>
</table>
| Change *Visible in Scoping* setting   | With restrictions:  
  - If the *Visible in Scoping* option is not selected, selection is allowed.  
  - If the *Visible in Scoping* option is selected, removal is not allowed. |
| Change *Visible in Fine-Tuning* setting | Yes     |
| Change *Mandatory* setting            | No      |
| Change *Scoping Question / Statement* | Yes     |
| Change *Anchor*                       | No      |
Maintenance of Business Configuration Objects

<table>
<thead>
<tr>
<th>Action</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete business configuration object</td>
<td>No</td>
</tr>
<tr>
<td>Change Name</td>
<td>No</td>
</tr>
<tr>
<td>Change Description</td>
<td>Yes</td>
</tr>
<tr>
<td>Change Create Code List Data Type setting</td>
<td>No</td>
</tr>
<tr>
<td>Change or delete fields</td>
<td>No</td>
</tr>
<tr>
<td>Add or delete new fields</td>
<td>Yes</td>
</tr>
<tr>
<td>Create (and delete) a new business configuration object</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Maintenance of Business Configuration Sets

<table>
<thead>
<tr>
<th>Action</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete business configuration set</td>
<td>No</td>
</tr>
<tr>
<td>Change Business Configuration Object Type</td>
<td>No</td>
</tr>
<tr>
<td>Change Name</td>
<td>No</td>
</tr>
<tr>
<td>Change Description</td>
<td>No</td>
</tr>
<tr>
<td>Change Business Configuration Object</td>
<td>No</td>
</tr>
<tr>
<td>Business configuration set field values: Change or delete rows</td>
<td>No</td>
</tr>
<tr>
<td>Business configuration set field values: Add or delete new rows</td>
<td>Yes</td>
</tr>
<tr>
<td>Create (and delete) a new business configuration set</td>
<td>Yes</td>
</tr>
</tbody>
</table>

See Also

- Business Configuration Quick Guide  [page 272]
- Create a Patch on a Customer's Tenant  [page 137]
- Create a Patch on a Development Tenant  [page 141]
- Lifecycle Management of Customer-Specific Solutions  [page 127]

6.3.3 Tasks

6.3.3.1 Create a Business Option

Overview

You must create at least one business option for your solution so that customers can activate the solution in the production environment. You anchor the business option in the business adaptation catalog (BAC) so that it is available in the SAP solution during scoping.

If you want to create more than one business option, you need to create a business topic first to group the business options. For more information, see Create a Business Topic and Business Options  [page 283].
Prerequisites

- You have created and opened a solution in the SDK.
- To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the Business Configuration work center. For more information, see User Setup Quick Guide [page 39].

Procedure

1. In the Solution Explorer, right-click the Business Configuration node and select Create BAC Elements. The Business Configuration Wizard opens.

2. In the Create and Anchor Elements in the Business Adaptation Catalog step, do the following:
   - Under Type of Business Configuration, select Business Option.
   - Under Basic Information, do the following:
     - Enter a name and a description for the business option and then enter a scoping question. Your scoping question indicates the functionality of the business option and allows key users to decide whether to select the business option during project implementation. For more information about project implementation, see Implementing a Project — First Implementation.
     - Optional: Select Visible in Fine-Tuning if you want to assign a business configuration set (BC set) to the business option and you want customers to be able to adapt the field values of the BC set during fine-tuning.
     - A fine-tuning activity is only mandatory for go-live in the customer system if a BC set that was created using a custom business configuration object (BCO) is assigned to the business option. In this case, you must also create a BC view. For more information about creating a BC view, see Create a Business Configuration View [page 295].
     - On the Overview tab, write a text that describes the function that this element provides and the benefits of using this element. Aim to write no more than three to five sentences. However, when additional valuable information is available, you should include it.
       - The Time Recording Integrated with Procurement business option allows you to use time recording for third party service agents, such as consultants and service technicians. The working times of service agents can then be allocated to procured services and purchase orders. The recorded times are taken into account in invoice verification.
     - On the Relevance tab, write a text that describes when or why the customer should select this element and what are the implications of selecting or not selecting this element. Explain the relevance of this element in terms of real-world business requirements.
       - The Time Recording Integrated with Procurement business option is relevant if you want to consider the working time recorded by service agents for invoice verification. If the Services with Time Sheet Recordings for Projects business option (within the Service and Non-Stock Material Procurement business topic) is selected, this business option is automatically included in your scope.
   - Under Business Adaptation Catalog (BAC), do the following:
Next to the Anchor field, click Select and anchor your business topic in the BAC by expanding the tree and selecting a business topic or business option group.

Optional: Next to the Required Scope field, click Select and then select the standard BAC elements in the SAP solution that you require for your solution by expanding the tree and selecting one or more business packages or business topics.

3. In the Review step, check that the information you have entered is correct and then click Finish. The system assigns all content that is relevant for business configuration such as business objects and business configuration sets (BC sets) to your business option. The content is displayed under the business option.

4. In the Solution Explorer, right-click the Business Configuration node and then select Activate to make the business option available in the BAC.

Result

The system has added the business option to the business adaptation catalog (BAC). You can check the result by logging on to the SAP solution as a business user. You select an implementation project in the Business Configuration work center and edit the project scope. The Scoping step displays the BAC elements and you can navigate to the business option you created.

See Also

Business Configuration Quick Guide [page 272]

6.3.3.2 Create a Business Topic and Business Options

Overview

You can create a business topic to group business options and provide multiple business options for customers to customize your solution.

Prerequisites

- You have created and opened a solution in the SDK.
- To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the Business Configuration work center. For more information, see User Setup Quick Guide [page 39].

Procedure

1. In the Solution Explorer, right-click the Business Configuration node and select Create BAC Elements. The Business Configuration Wizard opens.
2. In the Create and Anchor Elements in the Business Adaptation Catalog step, do the following:
   - Under Type of Business Configuration, select Business Topic and Business Options.
1. If you need to create only one business option, under Type of Business Configuration, select Business Option. For information about how to create one business option for a solution, see Create a Business Option [page 281].

- Under Basic Information, do the following:
  - Enter a name and a description for the business topic.
  - On the Overview tab, write a text that describes the function that this element provides and the benefits of using this element. Aim to write no more than three to five sentences. However, when additional valuable information is available, you should include it.

The Registered Products business topic allows you to track individual products, such as products with a serial number. These products are registered at the point-of-sale or when the customer calls with a support issue. Once entered in the system, this information is available for future interactions with the customer, for example for marketing and service delivery. The information tracked can include the individual product ID, warranty entitlements, product location, and configuration details. Once products are registered, they appear in the accounts fact sheet.

- On the Relevance tab, write a text that describes when or why the customer should select this element and what are the implications of selecting or not selecting this element. Explain the relevance of this element in terms of real-world business requirements.

The Registered Products business topic is relevant if you want to actively manage your customer installed base, especially if you want to track warranty entitlements against serialized products.

- Under Business Adaptation Catalog (BAC), do the following:
  - Next to the Anchor field, click Select and anchor your business topic in the BAC by expanding the tree and selecting a business package.
  - Optional: Next to the Required Scope field, click Select and select the standard BAC elements in the SAP solution that you require for your solution by expanding the tree and selecting one or more business packages or business topics. If you reference SAP content in BC sets, you need to select the SAP business topic to which the corresponding business option is assigned as the required scope.

3. In the Create Business Options and Assign Solution Content step, create your business options by doing the following:
   a. Click Create Business Option.
      The Create Business Option dialog box opens.
   b. In the Create Business Option dialog box, under Basic Information, do the following:
      1. Enter a name and a description for the business option.
      2. On the Overview tab, write a text that describes the function that this element provides and the benefits of using this element. Aim to write no more than three to five sentences. However, when additional valuable information is available, you should include it.

The Time Recording Integrated with Procurement business option allows you to use time recording for third party service agents, such as consultants and service technicians. The working times of service agents can then be allocated to procured services and purchase orders. The recorded times are taken into account in invoice verification.
On the Relevance tab, write a text that describes when or why the customer should select this element and what are the implications of selecting or not selecting this element. Explain the relevance of this element in terms of real-world business requirements.

The Time Recording Integrated with Procurement business option is relevant if you want to consider the working time recorded by service agents for invoice verification. If the Services with Time Sheet Recordings for Projects business option (within the Service and Non-Stock Material Procurement business topic) is selected, this business option is automatically included in your scope.

c. Optional: If you want the business option to appear in the BAC, select Visible in Scoping and under Scoping, do the following:
   - If your business option provides functionality that is essential for your solution, select Mandatory and enter a scoping statement that describes the functionality of the business option. By default, the business option will be in scope when a customer selects the business topic in the BAC.
   - If your business option is not mandatory, enter a scoping question. Your scoping question indicates the functionality of the business option and allows key users to decide whether to select the business option during project implementation.

d. Optional: Select Visible in Fine-Tuning if you want to assign a business configuration set (BC set) to the business option and you want customers to be able to adapt the field values of the BC set during fine-tuning; then under Fine-Tuning, do the following:
   1. If customers must complete a fine-tuning activity before the solution goes live, select Mandatory for Go-Live.
   2. Enter a description for the fine-tuning activity.

   A fine-tuning activity is only mandatory for go-live in the customer system if a BC set that was created using a custom business configuration object (BCO) is assigned to the business option. In this case, you must also create a BC view. For more information about creating a BC view, see Create a Business Configuration View [page 295].

e. Under Anchor Business Option below, you can either accept the selection of Header Element of the Solution or select SAP Element and then Select Anchor to anchor your business option in the BAC. Expand the tree and select your business topic, an SAP business topic or business option group and click OK.

f. Repeat the steps above for each business option and click Next.
The Create Business Options and Assign Solution Content step now displays all business options that you created on the right.

4. In the Create Business Options and Assign Solution Content step, assign solution content to each business option by using the arrow buttons or by means of drag-and-drop. The system displays the solution content you assign under the business option. You can also use the arrow buttons or drag-and-drop to unassign solution content.

   You must assign all solution content such as business objects and BC sets to a business option before you deploy your solution. If dependencies exist between BC sets defined in your solution, for example, a value defined in one BC set is referenced in a second BC set, you must assign both BC sets to the same business option or you must assign the referenced BC set to a mandatory business option.

5. In the Review step, check that the information you have entered is correct and then click Finish.
6. In the Solution Explorer, right-click the Business Configuration node and then select Activate to make the business topic available in the BAC.

Result

The system has added the business topic to the business adaptation catalog (BAC). You can check the result by logging on to the SAP solution as a business user. You select an implementation project in the Business Configuration work center and edit the project scope. The Scoping step displays the BAC elements and you can select the business topic that you created.

If you have created all business configuration content your solution requires, you can trigger deployment of your business configuration content to be able to use the content in your solution. To do this, right-click the Business Configuration node and select Deploy Business Configuration.

When you use the Deploy Business Configuration function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the Deploy Business Configuration function deploys all business configuration content independently of any business options you have created.

See Also

Business Configuration Quick Guide [page 272]

6.3.3.3 Create a Country and Business Options

Overview

You can define a country and create one or more business options if your solution requires country-specific scoping for a country that is not supported by the standard SAP solution.

Prerequisites

- You have created and opened a solution in the SDK.
- To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the Business Configuration work center. For more information, see User Setup Quick Guide [page 39].

Procedure

1. In the Solution Explorer, right-click the Business Configuration node and select Create BAC Elements. The Business Configuration Wizard opens.
2. In the Create and Anchor Elements in the Business Adaptation Catalog step, do the following:
   a. Under Type of Business Configuration, select Country and Business Options.
   b. Under Basic Information, enter a name and description and then select the country for which your solution is relevant.
3. In the *Create Business Options and Assign Solution Content* step, create one or more business options for your country by doing the following:
   
   a. Click *Create Business Option*. The *Create Business Option* dialog box opens.
   
   b. In the *Create Business Option* dialog box, under *Basic Information*, do the following:
      
      ● Enter a name and a description for the business option.
      
      ● On the *Overview* tab, write a text that describes the function that this element provides and the benefits of using this element. Aim to write no more than three to five sentences. However, when additional valuable information is available, you should include it.

      ![The Time Recording Integrated with Procurement business option allows you to use time recording for third party service agents, such as consultants and service technicians. The working times of service agents can then be allocated to procured services and purchase orders. The recorded times are taken into account in invoice verification.](image)

      ● On the *Relevance* tab, write a text that describes when or why the customer should select this element and what are the implications of selecting or not selecting this element. Explain the relevance of this element in terms of real-world business requirements.

      ![The Time Recording Integrated with Procurement business option is relevant if you want to consider the working time recorded by service agents for invoice verification. If the Services with Time Sheet Recordings for Projects business option (within the Service and Non-Stock Material Procurement business topic) is selected, this business option is automatically included in your scope.](image)

   c. Optional: If you want the business option to appear in the BAC, select *Visible in Scoping* and under *Scoping*, do the following:
      
      1. If your business option provides functionality that is essential for your solution, select *Mandatory* and enter a scoping statement that describes the functionality of the business option. If your business option is not mandatory, enter a scoping question. Your scoping question indicates the functionality of the business option and allows key users to decide whether to select the business option during project implementation.

      ![When you define a country, you must create at least one business option that is visible in scoping.](image)

      2. Click *Select Anchor* and anchor your business option in the BAC by expanding the tree and selecting a business topic or a business option group.

   d. Optional: Select *Visible in Fine-Tuning* if you want to assign a business configuration set (BC set) to the business option and you want customers to be able to adapt the field values of the BC set during fine-tuning; then under *Fine-Tuning*, do the following:
      
      1. If customers must complete a fine-tuning activity before the solution goes live, select *Mandatory for Go-Live*.

      2. Enter a description for the fine-tuning activity.

      ![When you define a business configuration set (BC set), you must also specify whether key users are able to add or delete the values of specific fields. For more information about creating a BC set, see *Create a Business Configuration Set* [page 289].](image)

   e. Repeat the steps above for each business option.
The **Create Business Options and Assign Solution Content** step now displays all business options that you created for the country on the right.

4. In the **Create Business Options and Assign Solution Content** step, assign solution content to each business option by using the arrow buttons or by means of drag-and-drop.

   The system displays the solution content you assign under the business option. You can also use the arrow buttons to unassign solution content.

   ! You must assign all solution content such as business objects and BC sets to a business option before you deploy your solution. If dependencies exist between BC sets defined in your solution, for example, a value defined in one BC set is referenced in a second BC set, you must assign both BC sets to the same business option or you must assign the referenced BC set to a mandatory business option.

   You must assign all solution content such as business objects and BC sets to a business option before you deploy your solution. If dependencies exist between BC sets defined in your solution, for example, a value defined in one BC set is referenced in a second BC set, you must assign both BC sets to the same business option or you must assign the referenced BC set to a mandatory business option.

5. In the **Review** step, check that the information you have entered is correct and then click **Finish**.

6. In the **Solution Explorer**, right-click the **Business Configuration** node and then select **Activate** to make the country and its business options available in the BAC.

7. Check that the system has added the country and the business options you created to the business adaptation catalog (BAC) by doing the following:

   a. Log on to the SAP solution as a business user.

   b. In the **Business Configuration** work center, open the **Implementation Projects** view.

   c. Select a project and click **Edit Project Scope**. The **Edit Project Scope** guided activity is displayed.

   d. In the **Country and Type of Business** step, click **Edit Countries** and check that the country you created is display in the list of available countries.

   e. In the **Implementation Focus** step, select **Complete Solution**.

   f. In the **Scoping** step, check that the visible business options you created using the **Business Configuration Wizard** are displayed.

   The business options are displayed in the **Scoping Element** column under the business topic you selected as an anchor. The description and scoping questions you defined in the **Business Configuration Wizard** are displayed in the **Questions** step.

**Result**

The system has added your country and the business options you created to the business adaptation catalog (BAC) and you have checked the result in the SAP solution. The country description is displayed in the business adaptation catalog in the **Country and Type of Business** step.

If you have created all business configuration content your solution requires, you can trigger deployment of your business configuration content to be able to use the content in your solution. To do this, right-click the **Business Configuration** node and select **Deploy Business Configuration**.

```
When you use the **Deploy Business Configuration** function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the **Deploy Business Configuration** function deploys all business configuration content independently of any business options you have created.
```
6.3.3.4 Create a BC Set with an Implicit BCO

Overview
You can create a business configuration set (BC set) with an implicit business configuration object (BCO) to configure the behavior of business processes in your solution and define code list data types that you can use to model fields in a business object.

Prerequisites
You have created and opened a solution in the SDK.

Procedure

1. In the Solution Explorer, right-click the Business Configuration node and select Create Business Configuration Set.
   The Business Configuration Set Wizard opens.
2. In the Select a BCO Type and Create a BC Set step, do the following:
   b. Under Basic Information, enter a name and a description; then click Next.
3. Do the following:
   a. In the Define Parameters of the Business Configuration Set step, add the parameters that you require by entering a name and a description for each parameter.
   b. In the Define Parameter Values and Description step, create the values associated with each parameter by doing the following:
      1. From the Parameter list, select a parameter.
      2. Enter one or more values and a description for each value.
      3. Repeat the steps above for each parameter in the list.
4. In the Review step, check that the information you have entered is correct and then click Finish.
   A new item with the file extension .bcc appears in the Solution Explorer under the Business Configuration node.
   The name of the business configuration objects (BCO) and the data types that are generated during activation are displayed in the Review step.
5. In the Solution Explorer, right-click the .bcc file and select Activate.
   The system generates the required business configuration object(s) and data type(s).
6. In the Business Configuration Wizard, in the Create Business Options and Assign Solution Content step, assign your BC set to a business option by using the arrow buttons or by means of drag-and-drop.
For more information about assigning solution content to business options, see Create a Business Topic and Business Options [page 283].

7. In the Solution Explorer, right-click the Business Configuration node and select Deploy Business Configuration to be able to use your business configuration content in a solution.

The system deploys the business configuration content you created and makes the parameter values available in the development environment.

When you use the Deploy Business Configuration function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the Deploy Business Configuration function deploys all business configuration content independently of any business options you have created.

Result
You have created a BC set to your solution and assigned it to a business option. You can use the generated list data type, for example, to model fields in a business object.

See Also
Business Configuration Quick Guide [page 272]
Create a BC Set Using an SAP BCO [page 290]
Create a BC Set Using a Custom BCO [page 293]

6.3.3.5 Create a BC Set Using an SAP BCO

Overview
You can create a business configuration set (BC set) using an SAP business configuration object (BCO) to configure the behavior of standard business processes in the SAP solution. If you create an implementation of an enhancement option which is for single use only, you can also define a BC set using an SAP BCO so that you can enhance the filter values.

For customer-specific solutions:
- You can only create a BC set using an SAP BCO if you are developing the solution on a development tenant. For information, see Lifecycle Management of Customer-Specific Solutions on a Development Tenant [page 139].
- You cannot create BC sets using SAP BCOs in a solution template.

Prerequisites
- You have created and opened a solution in the SDK.
You have permission to access the business configuration objects (BCO) that are released with the SAP public solution model (PSM).

Procedure

1. In the Solution Explorer, right-click the Business Configuration node and select Create Business Configuration Set.

   The Business Configuration Set Wizard opens.

2. In the Select a BCO Type and Create a BC Set step, do the following:
   a. Under Business Configuration Object Type, select Use SAP Business Configuration Object.
   b. Under Basic Information, enter a name and a description.
   c. Under Business Configuration Object, in the search field, type the name of the SAP BCO you want to use or use the Search function to search for a BCO by deployment unit, namespace, or name.

   You can only use BCOs that are released with the SAP public solution model (PSM).

   Under Business Configuration Object, the system displays the deployment unit, the namespace, whether a fine-tuning activity is available, and a description of the BCO.

3. In the Define the Values of Your Business Configuration Set step, specify the field values of your BC set by doing the following:
   a. In the BCO hierarchy view, select the root node.
      If the BCO only has one node, by default, the hierarchy is hidden.
   b. In the field columns, specify values for at least all mandatory fields.
      An asterisk (*) is used to indicate mandatory fields. Field values must comply with the data type of the field. You can display information about a field by clicking the field name.
   c. Repeat the steps above for each node.

4. In the Review step, check that the information you have entered is correct and that the namespace of your solution (/BY*/) is added to the key field; then click Finish.

   A new item with the file extension .bcc appears in the Solution Explorer under the Business Configuration node. If you access the values you defined for the BCO using the SAP Business ByDesign scripting language, for example, you create a query, the key fields might contain the solution namespace as a prefix, which is also displayed in the Review step of the Business Configuration Set Wizard.

5. In the Solution Explorer, right-click your business configuration set and select Activate.

   The system generates the required business configuration object(s) and data type(s).

6. In the Business Configuration Wizard, in the Create Business Options and Assign Solution Content step, assign your BC set to a business option by using the arrow buttons or by means of drag-and-drop.

   For more information about assigning solution content to business options, see Create a Business Topic and Business Options [page 283].

7. In the Solution Explorer, right-click the Business Configuration node and select Deploy Business Configuration to be able to use your business configuration content in a solution.

   The system deploys the business configuration content you created and makes the field values available in the development environment.
When you use the **Deploy Business Configuration** function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the **Deploy Business Configuration** function deploys all business configuration content independently of any business options you have created.

**Result**

You have created a BC set using an SAP BCO and assigned it to a business option.

**See Also**

- Business Configuration Quick Guide  [page 272]
- Enhancement Implementations Quick Guide  [page 364]

### 6.3.3.6 Create a Business Configuration Object

**Overview**

You can create a business configuration object (BCO) and use it as the basis of a business configuration set (BC set).

**Prerequisites**

You have created and opened a solution in the SDK.

**Procedure**

1. In the **Solution Explorer**, right-click the **Business Configuration** node and select **Create Business Configuration Object**.
   
   The **Business Configuration Object Wizard** opens.

2. In the **Create Business Configuration Object** step, do the following:
   
   - Under **Basic Information**, enter a name and description.
   
   - Optional: Select **Create Code List Data Type** if you want the system to create a code list data type that can be used, for example, in business objects and business objects extensions. In this case, only one key field is allowed.

3. In the **Define Fields of the Business Configuration Object** step, for each field you define, enter the following data:
   
   - **Field name**
   
   - **Field description**
   
   - **Field type**
     
     Possible values are:
○ Key: key field of the business configuration object
○ Attribute: language-independent attribute of the business configuration object
○ Description: language-dependent attribute of the business configuration object

- **Data type**
  For more information about the supported data types and default values, see [Data Types](page 78).

- **Data subtype**
  If you define a field of data type *Code*, you must select a data subtype.

- **Length**
- **Decimal**
- **Mandatory**

4. In the **Review** step, do the following:
   a. Check that the information is correct:

<table>
<thead>
<tr>
<th>Name</th>
<th>[name of the business configuration object]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>[description of the business configuration object]</td>
</tr>
<tr>
<td>Data Type*</td>
<td>[name of the business configuration object]Code</td>
</tr>
</tbody>
</table>

   * The data type is displayed if you have selected the Create Code List Data Type option in step 2.

   b. Click **Finish**.
      A new item with the file extension `.bco` appears in the Solution Explorer under the Business Configuration node.

5. Activate the business configuration object by right-clicking the `.bco` file and selecting **Activate**.

**Result**

You have created a BCO that you can now use as the basis of a business configuration set (BC set). You can reference the values of this BCO in more than one BC set. For more information, see [Create a BC Set Using a Custom BCO](page 293).

**See Also**

- Business Configuration Quick Guide [page 272]
- Create a BC Set Using an SAP BCO [page 290]

**6.3.3.7 Create a BC Set Using a Custom BCO**

**Overview**

You can create a business configuration set (BC set) using a business configuration object (BCO) that you have defined in your solution.
Prerequisites

You have created and activated a business configuration object in your solution. For more information about creating a BCO, see Create a Business Configuration Object [page 292].

Procedure

1. In the Solution Explorer, right-click the Business Configuration node and select Create Business Configuration Set.
   The Business Configuration Set Wizard opens.

2. In the Select a BCO Type and Create a BC Set step, do the following:
   b. Under Basic Information, enter a name and a description.
   c. Under Business Configuration Object, in Name, select the business configuration object on which you want to base your BC set; then click Next.

   You can only use BCOs that you created in the solution.

3. In the Define the Values of Your Business Configuration Set step, specify the field values of your BC set.

   The field values must comply with the data type of the field. You can display information about a field by clicking the field name.

4. In the Review step, check that the information you have entered is correct and then click Finish.
   A new item with the file extension .bcc appears in the Solution Explorer under the Business Configuration node.

5. In the Solution Explorer, right-click your business configuration set and select Activate.

6. In the Business Configuration Wizard, in the Create Business Options and Assign Solution Content step, assign your BC set to a business option by using the arrow buttons or by means of drag-and-drop.
   For more information about assigning solution content to business options, see Create a Business Topic and Business Options [page 283].

7. In the Solution Explorer, right-click the Business Configuration node and select Deploy Business Configuration to be able to use your business configuration content in a solution.
   The system deploys the business configuration content you created and makes the field values available in the development environment.

   When you use the Deploy Business Configuration function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the Deploy Business Configuration function deploys all business configuration content independently of any business options you have created.
6.3.3.8 Create a Business Configuration View

Overview

You can create a business configuration view (BC view) for a business configuration set (BC set) to allow key users to add new values during fine-tuning or to delete the values of a BC set. You create the view in the studio and you model the view in the UI designer.

You can only create a BC view for a business configuration set (BC set) that was created using a custom business configuration object (BCO). You cannot create a BC view for an SAP BCO.

Prerequisites

- You have created a custom business configuration object (BCO). For information, see Create a Business Configuration Object [page 292]. When you create your BCO, if you want the system to create a code list, only one key field is allowed.
- To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the Business Configuration work center. For more information, see User Setup Quick Guide [page 39].

Procedure

1. In the studio, create a business configuration view by doing the following:
   a. In the Solution Explorer, activate the BCO for which you want to create the view by right-clicking the .bco file and selecting Activate.
   b. In the Properties window, make sure that the activation status of the BCO is Runtime objects up to date.
   c. Right-click the .bco file and select Create Business Configuration View. The Create Business Configuration View dialog box opens.
   d. Enter a name for the view, make sure that your BCO is selected, and then click OK. A new item with the file extension .QA.uicomponent appears in the Solution Explorer.
   e. In the Solution Explorer, open the BC view by right-clicking the .QA.uicomponent file and selecting Open in UI Designer. The view opens in the UI designer.
2. Optional: In the UI designer, modify the BC view, for example, change the field labels:
a. Change to edit mode. Select the column whose name you want to change and in the Property Explorer, under Text Information, click the Label field.

b. Click the arrow button and in the Dependent Property Editor, in the Overridden Text field, enter a new name.

3. In the UI designer, assign your BC view to the Business Configuration work center by doing the following:
   a. In the Configuration Explorer, open the Implementation Projects work center view by opening the /SAP_BYD_APPLICATION_UI/BCTools/WorkCentre/ folder. The file name of the view is ImplementationProjects_WCView.WCVIEW.uiwocview.
   b. In the Extensibility Explorer, select the ImplementationProjects_WCView.ao.uianchor anchor and click Assign Component to View.
   c. In the Type field, select Navigation Target.
   d. In the Component field, select your BC view from the Repository. It is stored in the following folder: [name of your solution]_BC/SRC/[your BC view]. Click Apply.

   You can find the name of your solution in the studio: In the Solution Explorer, select your solution. In the Properties, in the Project File field, there is the name of your solution.

e. Save, activate, and refresh your BC view.

4. In the studio, create a BC set by doing the following:
   b. Follow the wizard steps and make sure that:
      - Under Business Configuration Object Type, Business Configuration Object Defined in Solution is selected.
      - Under Business Configuration Object, the BCO you created is displayed in the Name field.
      - You define that key users can add and delete field values during fine-tuning. If you do not allow the addition nor the deletion of field values, customers will not be able to make any changes on the BC view.
   c. In the Solution Explorer, activate the BC set by right-clicking the .bco file and selecting Activate.
   d. In the Properties window, make sure that the activation status of the BC set is Runtime objects up to date.

   For information about creating a BC set, see Create a BC Set Using a Custom BCO [page 293].

5. Make your BC view available in the business adaptation catalog (BAC) by doing the following:
   b. Follow the wizard steps and create a business option or a business topic and more than one business option. Make sure that you select Visible in Fine-Tuning and, if appropriate, Mandatory for Go-Live for the business option to which you want to assign the BC set that you created to make your BC view accessible in fine-tuning.
   c. In the Solution Explorer, right-click the Business Configuration node and then select Activate to make the business option available in the BAC.

6. Open your BC view for testing in the SAP solution by doing the following:
   a. Log on to the SAP solution as a business user.
   b. In the SAP solution, in the Business Configuration work center, open the Implementation Projects view.
   c. Select a project and click Edit Project Scope.
The Edit Project Scope guided activity is displayed.

d. Follow the guided activity and answer the scoping question for your business option with Yes. The business topic or business option you created is displayed in the Scoping phase under Scoping Element. The description and scoping questions you defined are displayed in the Questions phase. When you finish the guided activity, the activity you assigned to your business option is added to the activity list.

e. In the Implementation Projects view, select the implementation project and click Open Activity List. The Activity List guided activity opens and displays the Prepare phase. Activity List displays all activities that are mandatory for go-live. You must add any activities that are optional manually. For information about adding optional activities, see Complete the Activity List under Add or Remove Optional Activities.

f. In the Activity List, click Hide Phased List and search for your activity by the activity description you defined in the studio.

g. Select your activity and click Open. Your BC view opens and you can test the result.

! If you have assigned for than one BC view to the same business option, a fact sheet with the links to all of the views is displayed when you click the activity.

**Result**

You have created a business configuration view and opened it in the SAP solution. You can now test the result, for example:

- Is the UI layout displayed as expected?
- Are the defined BC set values displayed as expected?
- Is the deletion of rows only possible for rows where the indicator was set in the studio?
- Can you create new rows?
- Is the creation of key fields with Z as a prefix prevented?
- Can you translate texts?
- Can you restore texts?

**See Also**

Business Configuration Quick Guide [page 272]

6.4 Business Objects

6.4.1 Business Objects Quick Guide

In the SDK, you can create your own business object and define its nodes and relevant elements in the business object definition. After you have defined your business object, you can generate the screens for your business object.
Business and Technical Background

Scripting Language
The scripting language is used to define business objects and business object extensions as well as to implement the business logic for SAP Business ByDesign partner solutions. This lightweight language is easy to learn and to use with its focused set of features.
For more information, see here [page 156].

Syntax for Business Object Definition
You use specific keywords and functions to describe the structure and interface of a business object in the SDK.
For more information, see here [page 158].

Queries
A query is a service operation that contains search parameters and fields from different business object nodes for use in search and information retrieval operations. It gives a semantic view of selection criteria.
For more information, see here [page 302].

Actions, Events and Validations
Actions, events and validations are script files that are used to implement the business logic for the nodes of a business object.
For more information, see here [page 301].

Syntax for Implementation of Business Logic
Methods and keywords enable you to implement the business logic for your solution in your script files in the SDK.
For more information, see here [page 178].

Reuse Libraries and Built-In Functions
The built-in functions and reuse libraries of the scripting language extend the scope and functions that you use to implement the business logic of your solutions.
For more information, see the following documentation:
- Built-In Functions [page 228]
- Basic Reuse Libraries [page 202]
- Reuse Libraries for Business Areas [page 223]

Introduction to Business Object Modeling
For more information, see here [page 70].
For an example on how to build a solution, see Example: Building a Solution [page 88].

Public Solution Model
In the SDK, you can explore SAP content and PSM documentation using the Repository Explorer tool window. For information, see Repository Explorer [page 53].

Development Guideline
The Development Guideline provides general programming rules and guidance on selected product standards.
Tasks

Create a Business Object

1. In the My Solutions tool window, open your solution.
2. In the Solution Explorer, right-click the project in your solution and select Add New Item.
3. In the Add New Item dialog box, select the Business Object template, enter a name for your business object, and click Add.

   If you use a template to create a new project item, do not change or delete the extension of the item name. For example, do not delete the .bo file extension of the business object file.

   A new item with the file extension .bo appears in the Solution Explorer. A document window opens that displays your business object file in a code editor.

Enter a Business Object Definition

1. Select the document window of your business object.

   In the code editor, the following information is displayed:
   
   - The import statement for the AP.Common.GDT SAP namespace that contains the standard data types:
     ```
     import AP.Common.GDT as apCommonGDT;
     ```
   
   You can import further namespaces so that you can use SAP business objects and data types when you define elements and associations.
   
   - The businessobject keyword and the name of your business object, for example, BonusRule.

   The names of the .bo file and of the business object must be identical. Do not change the name of the business object in the coding.

2. Define the nodes of your business object and their features, such as elements, actions, and associations with other nodes.

   In the following simple business object definition, the ID element is specified for the root node of the BonusRule business object:
   ```
   import AP.Comm.GDT as apCommonGDT;
   businessobject BonusRule {
      element ItemID : ID;
   }
   ```

   For more information about the keywords that you can use, see Syntax for Business Object Definitions [page 158].

3. In the File menu, click Save.
4. In the Solution Explorer, right-click your business object and select Activate.

   The content of your business object is activated.

Create a Query

In the SDK, you can define queries that allow you to access and combine the fields defined in different nodes or business objects in a single query and to combine data from different business object nodes. You can also use a query to make a custom business object available in the enterprise search.

For more information about creating a query, see here [page 307].
Define Access Control
In the SDK, you can implement instance-based access control for a business object you created by referencing an SAP business object for which an access context is defined.
For more information about defining access control, see here [page 309].

Assign Access Rights
If you have implemented instance-based access control for a business object you created in the SDK, you can restrict read and write access in the SAP solution on the basis of a work center views’s access context.
For more information about assigning access rights, see here [page 312].

Define the Business Logic for a Business Object
To define the business logic for your business object in the SDK, you use event, action and validation script files.
For more information, see here [page 313].

Generate Screens for a Business Object
In the SDK you can generate screens in the SAP solution based on your business object definitions, for example, work centers, floorplans and fact sheets.
For more information about generating screens for a business object, see here [page 314].

Add a Label and Tooltip to a Business Object
In the SDK you can generate screens in the SAP solution based on your business object definitions, for example, work centers, floorplans and fact sheets.
You can define a label and a tooltip for a business object field. If you then add the field to a screen:
- the label is displayed on the screen instead of the field name
- the tooltip is displayed when you place the cursor on the field
For more information about adding a label and tooltip for a business object, see here [page 316].

Create an OWL Excel Export
Object Work Lists (OWL) generated for custom business objects allow you to export the OWL data to a Microsoft Excel file. The file is based on a standard template, which you cannot change. However, you can format the generated spreadsheet as required.
For more information, see here [page 316].

Enable Change History
You can use an embedded component to ensure that a change history is written for all changes made to your custom business object. After adding the embedded component to your business object, you can display the change history in the SAP solution on the Changes tab of your work center. All changes such as add, create, and delete are displayed in a list.
For more information, see here [page 320].

See Also
Maintenance of Business Objects [page 305]
6.4.2 Business and Technical Background

6.4.2.1 Actions, Events, and Validations

Overview

Actions, events and validations are script files that are used to implement the business logic for the nodes of a business object.

Features

Action

An action is an element of a business object node that describes an operation performed on that node, for example, setting a status. The action operates on a set of node elements or instances that have been created or changed. Action script files are provided if you have added actions to your business object definition.

Event

An event is used to compute data that can be derived from other node elements, for example, totals of items and statuses, or from dependent objects, for example, AttachmentFolder or TextCollection. The script files for the AfterModify and BeforeSave events are provided by default:

- **AfterModify**
  This event determines the data of a node when the data has been changed. This is especially useful for derived data that needs to be updated immediately.

- **BeforeSave**
  This event determines the current data of a node.

- **AfterLoading** (for business objects only)
  You can implement this event to read and modify transient fields of the current node of a business object.

You cannot use an **AfterLoading** event to do the following:

- Read and modify transient fields of other nodes.
- Modify any persistent fields.
- Create or delete nodes.
- Call an action.

Validation

A validation is used to check if the conditions for saving a business object or a business object node are fulfilled. For example, you can implement a check so that the user cannot enter a value in a business object field that exceeds a given value. Validations can only read data and cannot change any values. The condition of a validation must always evaluate to a Boolean value, that is, to **true** or **false**. The validation script file **OnSave** is provided by default.
A validation script file that checks if the string "0243" is contained in the `elemData` field:

```csharp
if (this.elemData.Contains("0243")) {
    return(true);
} else {
    return(false);
}
```

For another example, see Example: Validate on Save [page 118].

**Mass enablement**

You can specify whether a script file is to be executed for multiple instances of business object nodes, which is referred to as mass enabling.

If you work with mass-enabled script files, consider the following:

- To mass enable a script file, you select the **Mass Enable** checkbox in the Create Script Files dialog box. For more information, see Define the Business Logic for a Business Object [page 313].
- If you want to modify a mass-enabled script file so that it can only be executed for single instances of business object nodes, you need to delete the script file first and then re-create it.

If you use the `this` keyword in the coding of a mass-enabled script file, `this` represents a collection of business object node references.

**See Also**

- Business Object Quick Guide [page 297]
- Syntax for Implementation of Business Logic [page 178]
- Define the Business Logic for a Business Object Extension [page 347]

### 6.4.2.2 Queries

#### Overview

A query is a service operation that contains search parameters and fields from different business object nodes for use in search and information retrieval operations. It gives a semantic view of selection criteria.

#### Default Query

When you activate a business object you created in the SAP Business ByDesign studio, by default, the system adds a `QueryByElements` query to all nodes of the business object. The query contains all elements of the node as search parameters. The query result is the instances of the node to which the `QueryByElements` query is attached. In general, the `QueryByElements` query has a linear dependency on the number of instances in the business object node (t = O(n), where n is the number of business object instances in the database).

The `QueryByElements` query is a simple query and does not provide full-text indexing. We therefore recommend that you use the default query when:

- The expected number of node instances or collections is small, for example, for an object with configuration data, or the execution frequency is low.
The selection parameter list contains an equal condition on an element that is the alternative key of the node. An alternative key is supported by an index so the runtime dependency is $t = O(\log n)$.

If you define an association in your business object as relevant for access control by using the `RelevantForAccessControl` annotation and you use the default `QueryByElements` query, error messages related to role-based access management (RBAM) might occur if you use the query to control access. These RBAM error messages are suppressed when you create a complex (custom) query in your solution.

Complex (Custom) Queries

A complex query is a custom query that allows you to access and combine the fields defined in different business object nodes or business objects in a single query and to combine data from different business object nodes. Complex queries support full-text indexing for all query elements ($t = O(\log n)$). We therefore recommend that you define a complex query when the expected number of records in the node is large.

In the SAP Business ByDesign studio, you can create queries using the `Query Wizard`. For information, see Create a Query [page 307]. When you define a query, you must specify which business object fields to include in the query. You can also specify which fields to:

- Include as search parameters
- Use in text search
- Display in the query result

If a [1..n] association is defined for a field, you cannot use the field in the query result. We therefore recommend that you select the most detailed node as the parent node. For more information about associations, see Syntax of Business Object Definitions [page 158].

You can access the query fields using the SAP Business ByDesign scripting language. For more information, see SAP Business ByDesign Scripting Language [page 156]. You can then include the query data in an object worklist (OWL) using the UI designer. For more information, see Model an Object Work List [page 518] and Configure a Query [page 549]. You must create a query to implement instance-based access control for the results displayed in the OWL. For information about defining access control, see Define Access Control [page 309].

See Also

Business Objects Quick Guide [page 297]

6.4.2.3 Screen Types

The studio enables you to create screens for your business object. In the `Create Screens` dialog, you are prompted to select the screen types you want to use. For information on how to create screens, see Generate Screens for a Business Object [page 314].

You can select the following screen types:
<table>
<thead>
<tr>
<th>Screen Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object Instance Floorplan (OIF)</strong></td>
<td>A floorplan that allows business users to create, view, and edit a business object. Typically, the OIF shows multiple tab pages. You can use this floorplan to model an application interface that allows the user to create, delete, display, and change the attributes and associations of business objects.</td>
<td><img src="image1" alt="Object Instance Floorplan" /></td>
</tr>
<tr>
<td><strong>Quick Activity Floorplan (QAF)</strong></td>
<td>A floorplan that allows business users to perform a specific task quickly. This can be self-contained or a short subtask within the context of a larger task (for example, a simple Create operation).</td>
<td><img src="image2" alt="Quick Activity Floorplan" /></td>
</tr>
<tr>
<td><strong>Fact Sheet (FS)</strong></td>
<td>A sheet that displays the most important properties of a business object on one page.</td>
<td><img src="image3" alt="Fact Sheet" /></td>
</tr>
<tr>
<td>Screen Type</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Object Work List (OWL)</td>
<td>A list pane that is always embedded in a work center view. The OWL contains business objects such as purchase orders or opportunities. Business users can, for example, search for, display, and edit business objects as well as trigger follow-on processes.</td>
<td><img src="image" alt="Object Work List" /></td>
</tr>
<tr>
<td>Object Value Selector (OVS)</td>
<td>A user interface pattern that supports business users when they select values for an input field. An OVS helps business users to select one item from a table of business object instances. If you choose Object Value Selector (OVS), a separate dialog box appears in which you can select the business object elements for the OVS.</td>
<td><img src="image" alt="Object Value Selector" /></td>
</tr>
<tr>
<td>Work Center View (WOCVIEW)</td>
<td>Part of a work center, for example, the Overview or Reports view.</td>
<td><img src="image" alt="Work Center View" /></td>
</tr>
</tbody>
</table>

### 6.4.2.4 Maintenance of Business Objects

**Overview**

A solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is Assembled. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer’s production tenant.
Change and Delete Restrictions

The following table gives an overview of the change and delete restrictions for business objects in a solution that is in maintenance mode. The system performs the change and delete restrictions checks when you save the business object.

<table>
<thead>
<tr>
<th></th>
<th>Adding Allowed</th>
<th>Deletion Allowed</th>
<th>Renaming Allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business object</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Business object node</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Business object node element</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Business object action</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Business object node association</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

The following changes are not allowed:
- Removing the using clause

association ToBonusPlan to BonusPlan using ID — > association
  ToBonusPlan to BonusPlan
  • Changing the target of an existing using clause

association ToBonusPlan to BonusPlan using ID — > association
  ToBonusPlan to BonusPlan using ID2

<table>
<thead>
<tr>
<th></th>
<th>Adding Allowed</th>
<th>Deletion Allowed</th>
<th>Renaming Allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type of element</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>AlternativeKey annotation on existing element</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>AlternativeKey annotation on new element</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Multiplicity of node</td>
<td>see comments</td>
<td>see comments</td>
<td>see comments</td>
<td></td>
</tr>
<tr>
<td>Multiplicity of association</td>
<td>see comments</td>
<td>see comments</td>
<td>see comments</td>
<td></td>
</tr>
<tr>
<td>Association valuation</td>
<td>see comments</td>
<td>see comments</td>
<td>see comments</td>
<td></td>
</tr>
<tr>
<td>Multiplicity of association valuation</td>
<td>see comments</td>
<td>see comments</td>
<td>see comments</td>
<td>Only the following multiplicity is allowed: [0..n]</td>
</tr>
</tbody>
</table>

The following changes are not allowed:
- [0..1] —> [1..1] or [1..n]
- [0..n] —> any
- [1..n] —> [0..1] or [1..1]
<table>
<thead>
<tr>
<th></th>
<th>Adding Allowed</th>
<th>Deletion Allowed</th>
<th>Renaming Allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association valuation condition</td>
<td>Yes</td>
<td>Yes</td>
<td>not applicable</td>
<td>Changing the conditions within a valuation expression is allowed.</td>
</tr>
<tr>
<td>Messages</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Business object query</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
| Assignment of deployment unit | Yes           | Yes              | Yes              | If associations to business objects in the deployment unit are defined, changes to the assignment cause an error due to a cross-deployment unit association.  
If an integration scenario has been defined, this has to be adapted.  |

See Also

- Business Objects Quick Guide  [page 297]
- Lifecycle Management of Customer-Specific Solutions  [page 127]

6.4.3 Tasks

6.4.3.1 Create a Query

Overview

In the SDK, you can define queries that allow you to access and combine the fields defined in different nodes or business objects in a single query and to combine data from different business object nodes. You can also use a query to make a custom business object available in the enterprise search.

You can use these queries to include the query data in an object worklist (OWL) and to access information from different nodes and business objects using the scripting language. You must create a query to implement instance-based access control for the results displayed in the OWL.

If you define an association in your business object as relevant for access control by using the RelevantForAccessControl annotation and you use the default QueryByElements query, error messages related to role-based access management (RBAM) might occur if you use the query to control access. These RBAM error messages are suppressed when you create a complex (custom) query in your solution.

Prerequisites

- You have created and opened a solution in the SDK.
You have created and activated at least one custom business object in your solution. For information, see Business Objects Quick Guide [page 297].

To make a custom business object available in Enterprise Search:
- The custom business object must contain an alternative key field, a description field, and a date/date time field.
- You must create a screen for the custom business object that is used to navigate to from the result list of the Enterprise Search. For example, you can create a Quick Activity (QA) floorplan.

For more information, see Generate Screens for a Business Object [page 314].

Procedure

1. In the Solution Explorer, right-click your business object and select Create Query. The Query Wizard opens.
2. In the Create a Query step, under Basic Information, define your query by doing the following:
   a. Select a parent node to link to the query. The parent node is the starting point for the field selection in the Select Query Fields step. If a [1..n] association is defined for a field, you cannot use the field in the query result. Therefore, we recommend that you select the most detailed node as the parent node. For more information about associations, see Syntax of Business Object Definitions [page 158].
   b. Enter a query node name.
   c. Optional: Enter a description.
   By default, the query name is QueryByElements.
3. Optional: If you want to make your business object available in search queries, under Basic Information, select Include in Enterprise Search; then do the following:
   a. Under Enterprise Search Details, define the following fields:
      ● Key Field
         The field of the business object to be used in search queries to sort or filter search results by ID.
         The alternative key fields of the custom business object can be used.
      ● Description Field
         The field of the business object to be used in search queries to filter search results by description.
         The text fields of the anchor node of the query or the subnodes of the anchor node of the query can be used.
      ● Last Changed Field
         The field of the business object to be used in search queries to sort or filter search results by last change date. The Date/Date Time fields of the custom business object can be used.
   b. Under UI Navigation, define the screen you want to open when an end user selects this business object in the search results. To do this, in Navigation Target UI, specify the target screen, for example, a quick activity floorplan (QA), and then specify the operation.
4. In the Select Query Fields step, select the fields you want to include in the query.
5. In the Define Query Parameters step, refine the properties of each query parameter.
   a. Optional: Change the parameter field name.
   b. Optional: Select the Selection checkbox if you want to include the parameter as a search parameter.
   c. Optional: Select the Basic Find checkbox if you want the parameter to be used in a text search.
   d. Optional: Select the Result checkbox if you want the parameter to be displayed in the query result.
You cannot use fields with [1..n] associations in a query result.

6. In the **Review** step, check that the information you have entered is correct and then click **Finish**. A new item with the file extension `.qry` appears in the **Solution Explorer**.

7. In the **Solution Explorer**, right-click the query you have created and select **Activate**.

**Result**

You have created a custom query in your solution. You can now use the scripting language to:

- Access your query
- Set the query parameters in order to be able to return a list of business object instances based on these parameters
- Execute the query

For more information see, **Scripting Language** [page 156].

You can create an object work list (OWL) and use the UI designer to design the screen to use the business object query data. For more information, see **Model an Object Work List** [page 518] and **Configure a Query** [page 549].

If you also used the query to make a custom business object available in the enterprise search, the custom business object is available in the enterprise search once the target UI has been assigned to a user. For information about assign access rights, see Business Users Quick Guide.

**See Also**

**Define Access Control** [page 309]

6.4.3.2 **Define Access Control**

**Overview**

In the SDK, you can implement instance-based access control for a business object you created by referencing an SAP business object for which an access context is defined.

Access rights in the SAP solution are based on work center and work center views. For each view, an access context is predefined that defines for which business data key users can restrict user access rights, for example, by company, site, sales, or employee. For example, if business users have access rights for a work center view but should only see company-specific data, you can define your business object so that it inherits the access control list (ACL) from an SAP business object that is assigned the company access context.

**Prerequisites**

- An SAP access context exists that you can use for your use case.
  
  For more information about access contexts and access restrictions, see User and Access Management under **Access Restrictions**.
An SAP business object exists with the necessary access context and an access control list (ACL) is directly assigned to this business object.

Your solution is a global solution.

Procedure

1. In the SDK, define a business object with an association that is marked as relevant for access control by doing the following:
   a. Create a business object, for example, with the name `AccessControlByCompany`, and enter your business object definition. For information about creating and defining a business object, see Business Objects Quick Guide [page 297].
   b. Add an element to the business object root node that you use to find the relevant business object instance.
      
      You want the access context to be controlled by company, so you define a `Company` element of type `ID`:
      ```
      element Company : ID;
      ```
   c. Add an association to the business object that is used for access control and mark the association with the annotation `[RelevantForAccessControl]`.
      
      For each business object, you can only use one association that is relevant for access control.
      - The source node of the association must be the root node.
      - The target node of the association must be root node of the target business object, that is, the controlling business object.
      - The target business object must have an association that is relevant for access control.
      - The target of this association must be the `AccessControlList` dependent object.
      ```
      [RelevantForAccessControl] association For_Access_Control to CompanyFinancialsProcessControl;
      ```
   d. Add a status element to the business object root node to handle errors, typically
      ```
      element Consistent : ConsistencyStatusCode;
      ```

2. In the SDK, create a query using the Query Wizard to include the query data in an object worklist (OWL). For information about creating a query, see Create a Query [page 307].
   ```
   For correct instance handling, a query based on Fast Search Infrastructure (FSI) is necessary. By default, the query includes the ACL data even if you do not select this information.
   ```

3. Create an Event-BeforeSave.abs1 script file and define an association for access control. For information about creating script files, see Define the Business Logic for a Business Object [page 313].
   a. In the Event-BeforeSave.abs1 script file, add the query you created to get the instance for access control.
Query = CompanyFinancialsProcessControl.QueryByCompany;
Selection = QueryCreateSelectionParams();
Selection.Add(Query.CompanyID, “I”, EQ, this.Company.Content);
QueryResults = Query.Execute(Selection);
In this example, access is controlled by company and the query uses the company at the root node as the basis for selection.

b. Find the relevant instance in the query and set the association to this instance.

    foreach (companycontrol in QueryResults) {
        this.For_Access_Control = companycontrol;
        break;
    }

c. For error handling, set a status on the business object root node.

    if (this.For_Access_control.IsSet ()){
        this.Consistent = “3”;
    } else{
        this.Consistent = “2”;
    }
In this example, the consistency status codes are “2” for consistent and “3” for inconsistent.

4. Assign the access context to the work center view.
   a. In the SDK, generate the screens for your business object.
      For information about generating screens, see Generate Screens for a Business Object [page 314].
   b. Open the work center view for editing by right-clicking the *WCVIEW.uiwocview screen file in the Solution Explorer and selecting Open in UI Designer.
      The view opens in the UI designer.
   c. In the UI designer, change to edit mode by clicking the Display <-> Edit button.
   d. In the Properties window, under RBAM Data, select Access Context Code and assign the same access context code that is assigned to the associated SAP business object.
   e. Save and activate your changes.
   f. Check that the correct screens are assigned: Under RBAM Data, select Assigned Objects.
      The system displays the directly linked OWL and other linked screens that you can navigate to from the OWL, for example, an object instance floorplan (OIF) or a quick activity floorplan (QAF).

5. Assign the business object that controls access to the linked screens by doing the following:
   a. In the SDK, double-click the screen file, for example, *QA.uicomponent or *OIF.uicomponent.
      The screen opens in the UI designer.
   b. In the UI designer, change to edit mode by clicking the Display <-> Edit button.
   c. In the Properties window, under RBAM Data, check that the following entries exist:
      • Access Check Mode is set to Privileged Except Access.
      • Authorization Classification Code is set to Application.
   d. Open the collection of the Access Controlled Business Object.
      The Select Business Objects dialog box opens.
   e. Check that the correct business object is selected and that the Unrestricted Access checkbox is not selected.
f. Save and activate your changes; then close the UI designer.

Result
You have defined an association for access control, defined the access context for the work center view, and defined access control for the screens for which you want to restrict access rights. You can now manage access rights for business users in the SAP solution. For information, see Assign Access Rights [page 312].

6.4.3.3 Assign Access Rights

Overview
If you have implemented instance-based access control for a business object you created in the SDK, you can restrict read and write access in the SAP solution on the basis of a work center views’s access context.

In the SAP solution key users can assign access rights to each employee who has a user in the system. For example, if an employee should only see data for a specific company and the company access context is assigned to a work center view, the key user can assign or restrict access to company-specific data for this view.

Prerequisites
You have defined associations that are relevant for access control in your business object and assigned the SAP business object that controls access to the work center views. For information, see Define Access Control [page 309].

You have a business user with the key user role for the SAP solution.

Procedure
1. In the SAP solution disable the access rights of all users for the relevant work center views.
   a. Open the Application and User Management work center \User and Access Management \Business User subview.
   b. Select the user whose access rights you want to assign using the access control settings you have defined for your business object; then click \Edit and choose the Access Rights option to open the access rights editor.
   c. On the Work Center and View Assignment tab, find your work center in the list of available work centers and clear the Assigned to User checkbox.
   d. Save your changes and log off from the system.
      Your changes are applied the next time you log on.
2. In the SDK update the authorization and access rights by doing the following:
   a. In the Solution Explorer right-click your solution and select Update Authorization and Access Rights.
      You must disable the access rights of all users in the SAP solution before you update authorization and access rights in the SDK.
3. Assign access rights to users for work centers views.
a. In the SAP solution open the **Application and User Management** work center  > **User and Access Management** subview.

b. Select the user whose access rights you want to edit; then click **Edit** and choose the **Access Rights** option to open the access rights editor.

c. On the **Work Center and View Assignment** tab, in the list of available work centers, find your work center and select the **Assigned to User** checkbox.

d. Save your changes.
   If you do not want to restrict any access rights, you can log off from the system. Your changes are applied the next time you log on.

For more information about assigning access rights to business users in the SAP solution, see Business Users Quick Guide under **Assign Access Rights**.

4. Restrict the access rights of users for work centers views.
   a. On the **Access Restrictions** tab, in the list of available work centers, find your work center and select the **Read and Write Access** checkbox.
   b. On the **Detailed Restrictions** tab, restrict read and write access on the basis of the work center views's access context, for example, based on the company.
   c. Save your changes and log off from the system.
   Your changes are applied the next time you log on to system.

For more information about access contexts and access restrictions in the SAP solution, see User and Access Management under **Access Restrictions**.

6.4.3.4 Define the Business Logic for a Business Object

**Overview**

To define the business logic for your business object in the SDK, you use event, action and validation script files. For more information, see Actions, Events, and Validations  [page 301].

**Prerequisites**

- You have saved the relevant business object.
- In order to define actions, you must have added them to your business object definition before. For more information, see the action  [page 159] keyword in Syntax for Business Object Definitions  [page 158].

**Procedure**

1. In the **Solution Explorer**, right-click your business object and select **Create Script Files**.
2. In the **Create Script Files** dialog box, for each of the business object nodes, select the events and validations that you want to implement. The system selects actions by default.
3. Optional: If you want script files to be mass-enabled, select the **Mass Enable** checkbox for the relevant actions, events or validations.
4. Click **OK**.

   The SDK generates script files in the Solution Explorer with the file extension `.absl`. The script files appear below the corresponding nodes of your business object. For example, `Event-AfterModify.absl` for an event, `Validation-OnSave.absl` for a validation, or `Action-Release.absl` for an action. Each script file opens in a separate document window.

5. In the code editor, select the script file you want to edit and enter the code for the event, validation, or action.

   For more information about the methods and keywords you can use, see **Syntax for Implementation of Business Logic** [page 178].

6. Save your script files.

See Also

- Built-In Functions [page 228]
- Basic Reuse Libraries [page 202]
- Reuse Libraries for Business Areas [page 223]
- Business Object Quick Guide [page 297]

6.4.3.5 Generate Screens for a Business Object

**Overview**

In the SDK you can generate screens in the SAP solution based on your business object definitions, for example, work centers, floorplans and fact sheets.

These screens then automatically contain an initial set of the fields that you have defined for your business object. The screens behave according to the business logic that you have implemented for your actions and events. You use the user interface designer to enhance the screens and adapt them to your needs.

You can also use your generated screens for mobile solutions. For more information, see Mobile Solutions.
Prerequisites

- If you generate screens based on an SAP business object, the business object must be released for the SAP public solution model (PSM).
- If you generate screens based on your business object, the business object must be activated. For more information, see Business Objects Quick Guide [page 297].

Procedure

1. In the Solution Explorer, do one of the following:
   - If you generate screens based on an SAP business object, proceed as follows:
     1. Right-click the project in your solution and select Add New Item.
     2. Select the Business Object template, enter a name and click Add.
   - If you generate screens based on one of your business objects, right-click the business object and select Create Screens.
2. In the Create Screens dialog box, enter a name for the new screen or set of screens.
   The system uses this name together with the abbreviation of the screen type as a file name, for example, SalesOrder_OWL/owl.uiocomponent.
3. Select the screen types you want to create or a whole floorplan scenario with navigation paths.
   You can select the following screen types:
   - Object Instance Floorplan (OIF)
   - Quick Activity Floorplan (QAF)
   - Fact Sheet (FS)
   - Object Work List (OWL)
   - Object Value Selector (OVS)
   - Work Center View (WOCVIEW)
   For detailed information on the various screen types, see Screen Types [page 303].
   If you select Floorplan Scenario with Navigation, the system generates the following screens and the navigation paths that you need to be able to navigate from one screen to the next screen:
   - Work Center
   - Work Center View
   - Object Instance Floorplan
   - Quick Activity Floorplan
   - Object Work List
4. Click OK.

Result

In the Solution Explorer, the files for the screens are listed. The system also creates a port type package file (.ptp file) that contains the navigation, for example, from the object work list to the quick activity floorplan.
To view a screen, right-click a screen file and select Preview Screen. To edit a screen, right-click a screen file and select Open in Ui Designer. For more information, see Introduction to the User Interface Designer [page 508].
### 6.4.3.6 Add a Label and Tooltip to a Business Object

**Overview**

You can define a label and a tooltip for a business object field. If you then add the business object field to a screen:

- the label is displayed on the screen instead of the field name
- the tooltip is displayed when you place the cursor on the field

**Procedure**

1. Create a business object element with a label and a tooltip using the following syntax:
   ```xml
   [Label ("<label name>")] [Tooltip (<tooltip name")]) element <element name> : <data type>;
   ```
   For more information, see Business Objects Quick Guide [page 297].
2. Generate screens for the business object. For more information, see here [page 314].

**Result**

In the Solution Explorer in the SAP Business ByDesign studio, right-click your screen and select Preview. The label is displayed for the business object field. The tooltip is displayed when you place the cursor on the field.

### See Also

- Label [page 165]
- Tooltip [page 168]

### 6.4.3.7 Create an OWL Excel Export

**Overview**

Object Work Lists (OWL) generated for custom business objects allow you to export the OWL data to a Microsoft Excel file. The file is based on a standard template, which you cannot change. However, you can format the generated spreadsheet as required.

**Prerequisites**

- You have opened your solution containing an active custom business object.
- You have created a floorplan scenario with navigation.
Procedure

2. Select you user and select [Edit ➤ Access Rights ➤ Edit ➤ Access Rights].
3. In the Work Center and View Assignment view, select the work center you have created and assign it to your user. Click Save and Close.
4. Log off from SAP Business ByDesign. Log on again and navigate to the work center you have assigned to your user. The work center you have created is displayed. It contains an Export button.
5. Click New and create several instances of your business object.
6. Complete the fields and select Save and then Close to get to the entry screen again.
7. On the OWL, click Go to view the data records you have created.
8. Click Export To Microsoft Excel.

Result

Microsoft Excel opens. The data records are displayed on a spreadsheet. You can save the data as it is or format it to suit your requirements.

See Also

Example: Export OWL Data to Excel [page 317]

6.4.3.8 Example: Export OWL Data to Excel

Overview

In this example, you create a solution with a custom business object. You generate a floorplan scenario with navigation and test the work center in SAP Business ByDesign. After that, you create data records, which you export to Microsoft Excel.

Procedure

1. Create a solution and generate the floorplan scenario.
   a. Log on to the SAP Business ByDesign studio and create a new solution.
   b. In the Solution Explorer, click the Add New Item button. The Add New Item dialog opens.
   c. Select Business Object, enter a name (for example, OWL_EXPORT.bo) and click Add. The Business Object Selection dialog opens.
   d. Enter the following code:
      ```java
      import AP.Common.GDT;
      businessobject OWL_EXPORT {
        [Label("Identifier")}] element ID:ID;
      ```
Save and activate the business object.

e. Right-click your solution and select Create Screens. The Create Screens dialog opens.

f. Select Floorplan Scenario with Navigation and click OK.

Create New Screens

The screens are generated and the corresponding files are displayed in the Solution Explorer.

2. **Assign the work center and perform a test.**

   a. Log on to SAP Business ByDesign and select Application and User Management > Business Users.

   b. Select your user and select Edit > Access Rights.

   c. In the Work Center and View Assignment view, select the work center you have created and assign it to your user. Click Save and Close.
Work Center Assignment

d. Log off from SAP Business ByDesign. Log on again and navigate to the work center you have assigned to your user.
The OWL_EXPORT work center is displayed. It contains an Export button.

e. Click New.

f. Create several instances of your business object. Complete the fields and select Save and then Close to get to the entry screen again.

g. On the OWL, click Go to view the data records you have created.
Data Records

h. Click **Export To Microsoft Excel**

**Result**

Microsoft Excel opens. The data records are displayed on a spreadsheet. You can save the data as it is or format it to suit your requirements.

**6.4.3.9 Enable Change History**

**Overview**

You can use an embedded component to ensure that a change history is written for all changes made to your custom business object. After adding the embedded component to your business object, you can display the change history in the SAP solution on the **Changes** tab of your work center. All changes such as add, create, and delete are displayed in a list.

**Prerequisites**

- You have created users with the Developer and Business User roles. For more information, see User Setup Quick Guide [page 39].
- You have created and opened a solution in the SDK.
- You have created a business object in your solution and entered a business object definition. For more information, see Business Objects Quick Guide [page 297].
- You have created an object instance floorplan (OIF) for your business object. For more information, see Generate Screens for a Business Object [page 314].
Procedure

1. **Add an embedded component to the object instance floorplan (OIF) you created for your business object.**
   a. In the SDK, in the Solution Explorer, double-click the object instance floorplan (OIF). The UI designer opens.
   b. In the UI designer, change to edit mode by clicking the Display <-> Edit button.
   c. On the Designer tab, click the Changes tab.
   d. In the Configuration Explorer, navigate to the ChangeDocuments_EC reuse component in the SAP_BYD_APPLICATION_UI/Reuse/ChangeHistory folder.
   e. Drag the reuse component to the top left pane of the OIF as shown in the graphic and then save your changes.

2. **Create an outport using the ECORootNodeID parameter.**
   a. On the Controller tab, right-click Outports and select Add Outport. The Outport Configuration: OutPort pane opens.
   b. Click the Add Parameter button.
   c. In the ParameterName field, enter ECORootNodeID and save your changes.

3. **Create a binding for the embedded component by binding the outport parameter and the ChangeHistory import parameters.**
   a. Next to ParameterBinding, click ... The Parameter Binding dialog box opens.
   b. Select NodeID and click OK.
   c. On the Designer tab, change to edit mode by clicking the Display <-> Edit button.
   d. On the Changes tab, click Bind. The Embedded Component Binding dialog box opens.
   e. Right-click Navigations and select Add Navigation. The Navigation Details are displayed.
   f. In the OutPort field, select OutPort and in the InPort field, select InECONodeID.
   g. Under OutPort Parameters and InPort Parameters, select the parameters and then click Bind. The outport and import parameters are displayed under Bound Parameters.
   h. Click OK and save your changes.

4. **Create an event handler to fire the outport.**
   a. On the Controller tab, right-click Event Handlers and select Add EventHandler. The new event handler is displayed under Event Handlers.
   b. Right-click the new event handler and select Rename; then enter a new name for the event handler.
   c. Under EventHandler Operations, click the Add Operation button.
d. Under **Type**, click in the field and then select **FireOutport**.

e. Under **Configure the operation**, select your outport and save your changes.

5. **To ensure that the user can open the embedded component by clicking the Changes tab, associate the new event handler with the **OnClick** event.**
   a. Click the **Properties** tab.
   b. On the **Designer** tab, click the **Changes** tab and then click the **Properties** tab again.
   c. On the **Properties** tab, under **Events** > **OnClick**, select the event handler you created and then save your changes.

6. **Specify the configuration parameters in the **ChangeDocuments** embedded component properties.**
   a. Select the embedded component and click the **Properties** tab.
   b. Under **Parameters**, specify the following parameters by clicking the arrow to the right of each field:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOName</td>
<td>&lt;Name of your business object&gt;</td>
</tr>
<tr>
<td>BONamespace</td>
<td>&lt;Namespace of your business object&gt;</td>
</tr>
<tr>
<td>ECOName</td>
<td>&lt;Name of your business object&gt;</td>
</tr>
<tr>
<td>ECONamespace</td>
<td>&lt;Namespace of your business object&gt;</td>
</tr>
</tbody>
</table>

   You can copy the name and namespace from the **BO Browser/Data Model** but copy and paste does not work. You can also copy and paste the namespace from the **Solution Properties** in the SDK.

c. Deselect the **IsAttributeSearchEnabled** option.

d. Save and activate the floorplan.

7. **Create a business configuration set in the SDK.**
   a. In the **Solution Explorer**, right-click the **Business Configuration** node and select **Create Business Configuration Set**.
      The **Business Configuration Set Wizard** opens.
   b. In the **Select a BCO Type and Create a BC Set** step, do the following:
      ● Under **Business Configuration Object Type**, select **Use SAP Business Configuration Object**.
      ● Under **Basic Information**, enter a name and description.
      ● Under **Business Configuration Object**, select **ChangeDocumentMandatoryObjectConfiguration**.
   c. In the **Define the Values of Your Business Configuration Set** step, specify the field values of your BC set by doing the following:
      1. Double-click in the field under **Object Type Code** and select your business object.
      2. Double-click in the field under **Business Object Node Element Operation Code** and select **Update**.
      3. Select the row with your entries and click **Copy Row**.
      4. In the copied row under **Business Object Node Element Operation Code**, select **Delete**.
      5. Click **Next** and then **Finish**.
   d. Activate your business configuration and then right-click the **Business Configuration** node and select **Deploy Business Configuration**.
      The system deploys the business configuration content you created and makes the field values available in the development environment.
When you use the **Deploy Business Configuration** function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the **Deploy Business Configuration** function deploys all business configuration content independently of any business options you have created.

8. **Test the embedded component.**
   a. Log on to the SAP solution as a business user and assign the work center that has the same name as your business object to your service agent.
   b. Navigate to the work center you created and click **Edit**.
   c. Enter data in one or more of the fields and save your changes.
   d. On the right of the screen, click **View All**.
   e. Select **Close** and **Refresh**.
      Your changes are displayed on the **Changes** tab.

![](image)

**Change history in SAP Business ByDesign**

**See Also**

- [Create a BC Set Using an SAP BCO](page 290)
- [Create an Embedded Component for a Custom Pane](page 473)
- [Add an Embedded Component to a Screen](page 474)

### 6.5 Analytics

#### 6.5.1 Analytics Quick Guide

Analytics in the SAP solution allows customers to support and monitor business processes and thus enables them to make informed decisions. The standard system provides predefined analytics content that is presented in reports. The data from key figures and characteristics in a data source or a key figure group serves as the basis of these reports. Key users can design reports that they can make available to other users. Users can then access these
reports from the Reports work center view of the corresponding work centers or from the Overview work center views.

In the SDK you can add data sources to a solution that key users for Analytics can use for reporting in the SAP solution. You can also create browser-based reports in the SDK and use the Analytics tools in the SAP solution to define the report layout and content.

**Business and Technical Background**

**Defining Analytics Content**

In the SDK you can define analytics content for custom business objects, for SAP business objects, or for a combination of both. In the SAP solution, customers use this content for reporting. Reports show the data and values derived from key figures and characteristics in data sources, key figure structures, and key figure grids.

For more information, see [here](#) [page 327].

**Data Sources**

A data source is an object that provides a multidimensional, analytical view of business data. Data sources are the basis of analytical report content and are associated with a specific access context or can be unrestricted.

For more information, see [here](#) [page 328].

**Combined Data Sources**

A combined data source combines data from two or more data sources that share common or similar characteristics.

For more information, see [here](#) [page 329].

**Joined Data Sources**

A joined data source joins data from two or more data sources that share common characteristics.

For more information, see [here](#) [page 330].

**Transformation Rules**

You can use transformation rules to calculate additional fields from business object fields. You mainly use transformation rules for date and date/time conversions. These transformed fields are added to the data source and can be used in reports.

For more information, see [here](#) [page 333].

**Aggregation**

To enable the calculation of key figures, the system can aggregate key figure values from a data source based on defined aggregation behavior. The system aggregates key figure values for multiple characteristics. The system can also aggregate key figure values for a specified characteristic; this is referred to as exception aggregation.

For more information, see [here](#) [page 334].

**Maintenance of Analytics Content**

A solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is Assembled. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer’s production tenant.

For information about creating a patch, see Patches for Customer-Specific Solutions [page 153].
For information about the change and delete restrictions for data sources in a solution that is in maintenance mode, see Maintenance of Analytics Content [page 335].

SAP Business ByDesign Tools for Analytics

To learn about how analytics functions are integrated in the SAP solution, see Overview of Analytics.

Tasks

Create a Data Source

In the SDK you can create a data source based on an SAP business object or a custom business object. A data source includes specific data such as key figures and characteristics that can be used for reporting in the SAP solution. For more information, see here [page 336].

Change a Data Source

1. In the SDK, open the solution that contains the data source you want to change.
2. In the Solution Explorer, double-click your data source.
   The data source opens in a document window and the information you defined using the Data Source Wizard is displayed on different tabs.
3. Make your changes and then click Save.
   You cannot change the entries in the mandatory fields (Namespace, Business Object, Anchor Node, and Data Source Name) on the Header Information tab.
   The system saves your changes.

   For information about the change and delete restrictions for data sources in a solution that is in maintenance mode, see Maintenance of Analytics Content [page 335].

Delete a Data Source

1. In the SDK, open the solution that contains the data source you want to delete.
2. In the Solution Explorer, right-click the data source and select Delete.
   If other project items depend on a data source, for example, if a data source is used in a report or a joined data source, you cannot delete this data source. To delete the data source, you must first delete any dependent project items.

   For information about the change and delete restrictions for data sources in a solution that is in maintenance mode, see Maintenance of Analytics Content [page 335].

Create a Combined Data Source

In the SAP solution you can use the Analytics tools to create a combined data source that combines data from multiple data sources.
For more information, see here [page 338].

Create a Joined Data Source

In the SAP solution you can use the Analytics tools to create a joined data source that joins data from multiple data sources that share common characteristics.
For more information, see here [page 339].
Create a Report

In the SDK you can create reports by using data sources that you have created. You can then define the report and the report layout in the SAP solution by using the Analytics tools.

For more information, see here [page 341].

Manage Reports

1. In the Solution Explorer, right-click your data source and select Manage Reports.
   The logon screen of the SAP solution opens in your browser.
2. Enter the user name and password that you entered when logging on to the repository.
   A report list opens and displays the reports assigned to your data source.
3. Optional: Change the content and layout of a report by doing the following:
   • To change the report definition, click Edit With and then select Report Wizard.
   • To change the report layout, click Edit With and then select Web Browser.

Change a Report Created in an Earlier Release

When you maintain a report that you created in an earlier release of your solution, you must ensure that the report has the same technical ID in all later releases to ensure that multiple versions of the report do not exist with different IDs and that all analytics content created by customers, for example, the field names of a data source, still has valid references. To change the technical ID, you must find out the technical ID that the system assigned to the initial version of your report and manually assign this ID to the report in the current release. To do this, do the following:

1. Open the initial version of your report in the development environment of the SAP solution and find the technical ID of the report by doing the following:
   a. In the Solution Explorer, right-click the data source on which your report is based and select Manage Reports.
      The logon screen of the SAP solution opens in your browser.
   b. Enter the user name and password that you entered when logging on to the repository with your Developer user role.
      The Design Reports view opens and displays the reports assigned to the data source including your report.
   c. Select your report, click Edit With and then select Report Wizard.
      The Report Wizard opens and displays the Define Report and Select Data Source step.
   d. At the top right of the screen, click Personalize and select This Screen.
      The Personalize panel opens.
   e. Under Sections, select Report and make sure that the Visible option is selected.
   f. Under Fields, find the Technical ID field and select the Visible option.
   g. Save your changes and close the Personalize panel.
      The screen now displays the technical ID of the report. Make a note of the ID.
2. Open your report in the current release of your solution and repeat the steps described above to personalize the screen of the Define Report and Select Data Source step.
3. Overwrite the ID displayed in the Technical ID field with the ID of the initial version of your report and save your changes by clicking Finish.
   The technical ID of the report is now the same in the initial release and in the current release of your solution.
6.5.2 Business and Technical Background

6.5.2.1 Defining Analytics Content

Overview

In the SDK you can define analytics content for custom business objects, for SAP business objects, or for a combination of both. In the SAP solution, customers use this content for reporting. Reports show the data and values derived from key figures and characteristics in data sources, key figure structures, and key figure grids.

Prerequisites

- You have created and opened a solution in the SDK.
- You understand how key users design reports in the SAP solution. For more information, see Overview of Analytics.
- You are assigned the Developer user role and have access rights for the Business Analytics work center. For more information, see User Setup Quick Guide [page 39].

Process Flow

1. You create a data source.
   - In the SDK, you add a data source to your solution and define the data source. For more information, see Create a Data Source [page 336].
     You must activate the data source to make it available in the SAP solution so that you can use it for your report content.
   - You use the Analytics tools in the SAP solution to create a combined data source or a joined data source. For more information, see Create a Combined Data Source [page 338] and Create a Joined Data Source [page 339].
2. In the SDK, you add a report to your solution.
   You must enable the key user mode so that the system saves all reports and report views you create in your solution.
   For more information, see Create a Report [page 341].
3. In the SAP solution, you define your report and the report layout in the Design Reports view of the Business Analytics work center.
   For more information, see Working with Reports Using the Report Wizard.
4. Optional: In the SDK, you create a work center and a work center view to which you want your report to be assigned; then, in the UI designer, you:
   a. Add the work center view to the work center.
   b. Assign a stable anchor to the work center view to which you want your report to be assigned.
      For more information, see Define an Anchor for a Floorplan [page 560] and Anchors [page 564].
      You must assign this anchor so that key users can assign reports to this work center view in the SAP solution.
c. Add the List and Gallery standard reports list views to your work center. These views list all reports that are assigned to the views of the work center. For more information, see Add a Report List View to a Work Center [page 559].

5. In a customer’s production system, an Analytics key user assigns the report to the work center view to make it available for other users.

   Only key users working in a customer’s production system can assign a report that you have created to a work center view. To assign a report to a work center view, the solution that contains the report must therefore be implemented in the customer’s production system. We recommend that you provide information about the work center views to which customers need to assign reports in your Solution Documentation. For more information, see Documentation Types in the SAP Solution [page 609].

See Also

Analytics Quick Guide [page 323]

6.5.2.2 Data Sources

Overview

A data source is an object that provides a multidimensional, analytical view of business data. Data sources are the basis of analytical report content and are associated with a specific access context or can be unrestricted. For information about access contexts, see User and Access Management under Access Restrictions.

Data sources contain the following objects:

- **Key figures**
  Fields according to which values are selected. Key figures are numeric values that have a unit of measure or currency assigned. Examples include Invoice Net Value and Purchase Order Quantity.

- **Characteristics**
  Fields according to which values are selected. Characteristics are alphanumeric, numeric, or text values. Examples include Product ID, Supplier, and Purchase Order Status. Variables are often associated with specified characteristics. Variables restrict characteristics to one or more specified value selections.

Working with Data Sources in the SAP Business ByDesign Studio

In the SDK you can create a data source based on an SAP business object or a custom business object. A data source includes specific data such as key figures and characteristics that can be used for reporting in the SAP solution.

When you define a data source, you select the relevant fields from your business objects, from SAP business objects, or from a combination of both. You can join the fields of multiple business objects by using associations. The system supports [1..1] associations with a foreign-key relationship.

Optionally, you can:

- Define transformation rules.
  For more information, see Transformation Rules [page 333].

- Reference other data sources.
Define aggregation rules to summarize key figures (Summation, Minimum, Maximum, Count) in a report using characteristics. For more information, see Aggregation [page 334].

The studio supports read access across deployment units in data sources. You can use cross-deployment-unit associations to access the data in business objects that are not assigned to the default deployment unit of your solution. For more information, see Syntax for Business Object Definitions [page 158] under Deployment Unit.

Working with Data Sources Using the SAP Business ByDesign Key User Tools

You can join or combine data from different data sources that share common characteristics and contain semantically different key figures. For more information, see Combined Data Sources [page 329] and Joined Data Sources [page 330].

Overview of SAP Data Sources

For information about the SAP data sources available in the different application areas, see Overview of Data Sources in SAP Business ByDesign. In addition to the available data sources, this information also gives an overview of the available key figure groups and common characteristics and key figures for data sources by area.

See Also

Analytics Quick Guide [page 323]
Create a Data Source [page 336]
Create a Combined Data Source [page 338]
Create a Joined Data Source [page 339]

6.5.2.3 Combined Data Sources

Overview

A data source is an object that provides a multidimensional, analytical view of business data. Data sources are the basis of analytical report content and are associated with a specific access context or can be unrestricted. A combined data source combines data from two or more data sources that share common or similar characteristics.
Use

In the SAP solution you can use the Analytics tools to create a combined data source that combines data from multiple data sources.

When you build a report using a combined data source, all characteristic values are displayed along with any key figure values.

You usually combine data sources that have the semantically similar characteristics. However, you can also combine data sources when the characteristics you require for a report are not available in one data source.

The following example shows a result set of a combined data source.

<table>
<thead>
<tr>
<th>ProductID</th>
<th>Supplier</th>
<th>Purchase Order ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCD-0001</td>
<td>1000010</td>
<td>1</td>
</tr>
<tr>
<td>MCD-0002</td>
<td>1000011</td>
<td>2</td>
</tr>
<tr>
<td>MCD-0022</td>
<td>1000012</td>
<td>3</td>
</tr>
<tr>
<td>MCD-0024</td>
<td>1000013</td>
<td>4</td>
</tr>
<tr>
<td>MCD-0005</td>
<td>1000014</td>
<td>5</td>
</tr>
</tbody>
</table>

Example of a Result Set of a Combined Data Source

See Also

Analytics Quick Guide  [page 323]
Create a Combined Data Source  [page 338]

6.5.2.4  Joined Data Sources

Overview

A data source is an object that provides a multidimensional, analytical view of business data. Data sources are the basis of analytical report content and are associated with a specific access context or can be unrestricted.

A joined data source joins data from two or more data sources that share common characteristics.
Use

In the SAP solution you can use the Analytics tools to create a joined data source that joins data from multiple data sources that share common characteristics.

When you build a report using a joined data source, the characteristic values that match in the joined characteristics in the data sources are displayed along with the values that are in the same row.

Joined data sources can have more than one join. A join is created when a characteristic is available in a data source that is an anchor. By default, the first data source you select is the anchor. The more joins there are in a joined data source, the more restricted the data is that is available for reports.

Join Types

There are different types of joins between data sources. You select a join type depending on your business requirements.

1. Inner Join

An inner join selects only those characteristic values from the data sources to be joined that match in the joined data source field along with any key figure values. Characteristic values in the joined data source fields that do not appear in both data sources are excluded from the joined result set.

The following example shows a result set of a joined data source that uses an inner join.
Example of a Result Set of a Joined Data Source That Uses an Inner Join

**Left Outer Join**

A left outer join selects all characteristics values from the data source that is selected as the anchor and those characteristic values that match in the data sources that are added along with any key figure values. By default, the first data source that you select is the anchor.

Setting the anchor to a different data source affects the values available because the joined data source fields may also change.

The following example shows a result set of a joined data source that uses a left outer join.
You can use transformation rules to calculate additional fields from business object fields. You mainly use transformation rules for date and date/time conversions. These transformed fields are added to the data source and can be used in reports.

Use

Business objects often contain date and date/time information, for example, for the release of an order. In reports, you normally want to aggregate data in time buckets, for example, to view the orders per month. For this purpose, you could define a `DateToMonth` or `DatetimeToMonth` transformation rule. As a result, the system adds a new field to the data source that contains only the month of the release `datetime` field. In a report, you can then easily aggregate the order data by using the month field.

You select transformation rules in the `Data Source Wizard`. For information about how to create a data source using the `Data Source Wizard`, see Create a Data Source [page 336].

Available Transformation Rules

Transformation rules are only available for fields of the data types `GlobalDateTime`, `Date`, and the `Name`, `Description`, and `Text` data types, which represent character strings.
## Transformation Rules for Date/Time Fields (GlobalDateTime Data Type)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatetimeToTime</td>
<td>Selects and retrieves only the time from a datetime field.</td>
</tr>
<tr>
<td>DatetimeToDay</td>
<td>Selects and retrieves only the day from a datetime field.</td>
</tr>
<tr>
<td>DatetimeToDate</td>
<td>Selects and retrieves only the date from a datetime field.</td>
</tr>
<tr>
<td>DatetimeToCalendarweek</td>
<td>Calculates the calendar week of a datetime field.</td>
</tr>
<tr>
<td>DatetimeToMonth</td>
<td>Selects and retrieves only the month from a datetime field.</td>
</tr>
<tr>
<td>DatetimeToQuarter</td>
<td>Calculates the quarter of a datetime field.</td>
</tr>
<tr>
<td>DatetimeToYear</td>
<td>Selects and retrieves only the year from a datetime field.</td>
</tr>
<tr>
<td>DatetimeToYearmonth</td>
<td>Selects and retrieves only the year and month from a datetime field.</td>
</tr>
<tr>
<td>DatetimeToYearquarter</td>
<td>Calculates the year and quarter of a datetime field.</td>
</tr>
<tr>
<td>DatetimesToDuration</td>
<td>Calculates the number of days between two datetime fields. *</td>
</tr>
</tbody>
</table>

## Transformation Rules for Date fields

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateToMonth</td>
<td>Selects and retrieves only the month from a date field.</td>
</tr>
<tr>
<td>DateToYearmonth</td>
<td>Selects and retrieves only the year and month from a date field.</td>
</tr>
<tr>
<td>DatesToDuration</td>
<td>Calculates the number of days between two date fields. *</td>
</tr>
</tbody>
</table>

## Transformation Rules for String Fields

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concatenate</td>
<td>Concatenates two string fields. *</td>
</tr>
</tbody>
</table>

* To use this transformation rule, you must select a second field.

## 6.5.2.6 Aggregation

### Overview

To enable the calculation of key figures, the system can aggregate key figure values from a data source based on defined aggregation behavior. The system aggregates key figure values for multiple characteristics. The system can also aggregate key figure values for a specified characteristic; this is referred to as exception aggregation.

### Features

The SDK supports the definition of standard aggregation. You can define exception aggregation behavior in the SAP solution using the Analytics tools.
Standard Aggregation
The standard aggregation behaviors you can define in the SDK are summation, minimum, maximum, and count. For example, you can define minimum and maximum aggregation for date key figures. You define aggregation behavior in the SDK using the Data Source Wizard. For information, see Create a Data Source [page 336].

Exception Aggregation
You define exception aggregation behavior for values of a calculated key figure. The deviation to the standard aggregation behavior is only valid in combination with a specified characteristic. You define exception aggregation behavior in the SAP solution using the Report Wizard. For information about different types of exception aggregation behavior in the SAP solution, see Exception Aggregation. For information about how to define exception aggregation behavior for a key figure using the Report Wizard, see Create and Edit a Key Figure.

See Also
Data Sources [page 328]

6.5.2.7 Maintenance of Analytics Content

Overview
A solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is Assembled. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer’s production tenant.

Change and Delete Restrictions
The following table gives an overview of the change and delete restrictions for analytics content in a solution that is in maintenance mode. The system performs the change and delete restrictions checks when you save the data source.

<table>
<thead>
<tr>
<th>Action</th>
<th>Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete a data source</td>
<td>No</td>
</tr>
<tr>
<td>Delete data source fields</td>
<td>No</td>
</tr>
<tr>
<td>Add or delete new data source fields</td>
<td>Yes</td>
</tr>
<tr>
<td>Change or delete transformations</td>
<td>No</td>
</tr>
<tr>
<td>Add or delete new transformations</td>
<td>Yes</td>
</tr>
<tr>
<td>Change the Display Name and Referenced Data Source fields of data source fields</td>
<td>No</td>
</tr>
<tr>
<td>Change or delete aggregations</td>
<td>No</td>
</tr>
<tr>
<td>Add or delete new aggregations</td>
<td>Yes</td>
</tr>
<tr>
<td>Create (and delete) a new data source</td>
<td>Yes</td>
</tr>
</tbody>
</table>
6.5.3 Tasks

6.5.3.1 Create a Data Source

Overview

In the SDK you can create a data source based on an SAP business object or a custom business object. A data source includes specific data such as key figures and characteristics that can be used for reporting in the SAP solution.

Prerequisites

- You have created and opened a solution in the SDK.
- To create a data source based on a custom business object, you must create and activate your custom business object.
  For more information, see Business Objects Quick Guide [page 297].
- If you want to allow referencing of your data source by another data source and enable navigation from a report to another floorplan that is not a standard UI, you must define this UI. For information, see Generate Screens for a Business Object [page 314].

Procedure

1. In the Solution Explorer, right-click your business object and select Create Data Source. The Data Source Wizard opens.

2. In the Create a Data Source step, define your data source by doing the following:
   - Under Basic Information:
     a. Select a namespace.
        By default, the namespace for your solution is selected; however, to create a data source based on an SAP business object, you can select an SAP namespace.
     b. Select a business object.
     c. Select an anchor node to link to the data source.
        The anchor node is the starting point for the field selection in the Select Data Source Fields step. If a [1..n] association is defined for a field, the field is not displayed in the business object hierarchy. Therefore, we recommend that you select the most detailed node as the anchor node. For more information about associations, see Syntax for Business Object Definitions [page 158] under Association.
     d. Enter the data source name.
     e. Optional: Enter a description of the data source.
     f. Optional: To allow access restrictions for data in reports based on this data source, select Enable Access Control.
        The access restrictions of the anchor business object are inherited; therefore customers can only assign reports based on this data source to work center views with the same access control.
g. Optional: To allow referencing by another data source and allow the fields of your data source to be used in reports based on the referencing data source, select Allow Referencing.

h. If you want to allow referencing, under Reference Details, select the key field.

i. Optional: Under Object Based Navigation, specify a target UI. Users will be able to navigate from the report to this floorplan.

3. In the Select Data Source Fields step, select the fields you want to include in the data source. You can include fields from other business objects by using associations. The SDK supports read access across deployment units in data sources. You can use cross-deployment-unit associations to access the data in business objects that are not assigned to the default deployment unit of your solution. For more information, see Syntax for Business Object Definitions [page 158] under Deployment Unit.

   Not all fields and associations modeled in the business object are supported. Only supported fields and associations are displayed.

4. Optional: In the Define Transformation Rules step, define transformation rules for fields of data type text, for example, ID and description, date, or GlobalDateTime by doing the following:
   a. In the Business Object Element area, select a field and on the Transformation tab, select a transformation rule.
   b. Define a field name for the transformation field and then press ENTER.
   c. If the transformation rule allows the selection of a second parameter, next to the Parameter field, click ...
      The Select Additional Transformation Rule Parameter dialog box opens.
   d. In the Select Additional Transformation Rule Parameter dialog box, select a second parameter by selecting a row and clicking the first cell in the row.

   For information about the available transformation rules, see Transformation Rules [page 333].

5. Optional: In the Define Referenced Data Sources step, modify the parameters by doing the following:
   a. Display and define the names for the selected fields that are characteristics.
   b. Assign an existing data source to each parameter.

   If necessary, you can change the display name for each parameter. You can assign an existing data source if the fields of an SAP business object are already included in an SAP data source.

6. Optional: In the Define Aggregation Behavior step, define the aggregation behavior for business object elements that are key figures.

   For more information about aggregation behavior, see Aggregation [page 334].

   The default aggregation of every key figure is Summation. You can change the default or add new aggregations.

7. In the Review step, check that the information you have entered is correct and then click Finish.

   If you have defined aggregations, check that the selected aggregations are displayed correctly.

A new item with the file extension .ds appears in the Solution Explorer.

8. In the Solution Explorer, right-click the data source you have created and select Activate.
Result

The data source is available in the SAP solution. You can check the result by logging on to the SAP solution. In the Business Analytics work center, navigate to the Design Data Source view and search for your data source.

In the SDK, you can now create a report that uses your data source. For more information, see Create a Report [page 341]. You define the report and the report layout in the SAP solution in the Design Reports view of the Business Analytics work center. For more information, see Working with Reports Using the Report Wizard.

See Also

Analytics Quick Guide [page 323]

6.5.3.2 Create a Combined Data Source

Overview

In the SAP solution you can use the Analytics tools to create a combined data source that combines data from multiple data sources.

You can create combined data sources in the Design Data Sources view of the Business Analytics work center. For information about combined data sources, see Combined Data Sources [page 329].

Prerequisites

- You are assigned the Developer user role and have access rights for the Business Analytics work center. For more information, see User Setup Quick Guide [page 39].
- You have enabled the key user mode so that the system saves the analytics content you create in your solution. To do this, in the Solution Explorer, right-click your solution and select Enable Key User.

Procedure

1. Log on to the SAP solution with the Developer user role.
2. In the Business Analytics work center, open the Design Data Source view and from New, choose the Combined Data Source option.
   By default, the combined data source contains the Counter key figure. The Counter key figure counts the number of instances of a specified object.
3. On the New Combined Data Source screen, click Add Data Source.
4. On the Add Data Source dialog screen, select a data source from the dropdown list.
   The screen displays the characteristics and key figures available in the selected data source. The initial selections serve as a basis for the combined data source. The characteristics and key figures are then displayed in the Combined Data Source Fields column.

You can only combine data sources that have compatible access contexts. For example, you can add a data source if the data source and the combined data source have a common access context. You can add any data source that has an unrestricted access context to a combined data source.
5. Select one or more additional data sources and add the data sources to the combined data source. The system checks whether the characteristics and key figures in the additional data sources are also available in the data source that you selected as a basis for the combined data source.

- If a characteristic or key figure is available in the combined data source, the system positions the characteristic or key figure in the same row of each data source.
- If a characteristic or key figure is not available in the combined data source, the system adds the characteristic or key figure to the combined data source.

Other Functions

- **Remove Data Source**
  The *Remove Data Source* dialog screen opens and displays the data sources you have added. To remove data sources, click the relevant checkboxes.

- **Add Row**
  Adds a row to which you can add a characteristic or key figure. The system then adds the characteristic or key figure to the relevant field type group.

- **Remove Row**
  Deletes the selected row.

  - The selected characteristic or key figure is only removed from the combined data source.

Result

The combined data source is available for reporting. You can create reports using data from the combined data source. For information about creating reports, see *Create a Report*  [page 341].

See Also

Analytics Quick Guide  [page 323]

6.5.3.3 Create a Joined Data Source

Overview

In the SAP solution you can use the Analytics tools to create a joined data source that joins data from multiple data sources that share common characteristics.

You can create joined data sources in the *Design Data Sources* view of the *Business Analytics* work center. For information about joined data sources, see *Joined Data Sources*  [page 330].

Prerequisites

- You are assigned the Developer user role and have access rights for the *Business Analytics* work center. For more information, see *User Setup Quick Guide*  [page 39].
You have enabled the key user mode so that the system saves the analytics content you create in your solution. To do this, in the Solution Explorer, right-click your solution and select Enable Key User.

**Procedure**

1. Log on to the SAP solution with the Developer user role.
2. In the Business Analytics work center, open the Design Data Source view and from New, choose the Joined Data Source option.
   By default, the joined data source contains the Counter key figure. The Counter key figure counts the number of instances of a specified object.
3. On the New Joined Data Source screen, click Add Data Source.
   You cannot join more than five data sources.
4. On the Add Data Source dialog screen, select a data source from the dropdown list.
   The screen displays the characteristics and key figures available in the selected data source. The initial selections serve as a basis for the joined data source. The characteristics and key figures are then displayed in the Joined Data Source Fields column.
   By default, the first data source you select is the anchor.
5. Select one or more additional data sources and add the data sources to the joined data source.
   The system checks whether the characteristics and key figures in the additional data sources are also available in the data source that you selected as a basis for the joined data source.
   If a characteristic or key figure is available in the joined data source, the system positions the characteristic or key figure in the same row of each data source.
   If a characteristic or key figure is not available in the joined data source, the system adds the characteristic or key figure to the joined data source.
   If a characteristic is available in a data source that is an anchor, the system creates a join. The more joins there are in a joined data source, the more restricted the data is that is available for reports.

**Other Functions**

- **Remove Data Source**
  The Remove Data Source dialog screen opens and displays the data sources you have added. To remove data sources, click the relevant checkboxes.
- **Add Row**
  Adds a row to which you can add a characteristic or key figure. The system then adds the characteristic or key figure to the relevant field type group.
- **Remove Row**
  Deletes the selected row.

  The selected characteristic or key figure is only removed from the joined data source.
- **Check**
  Checks whether the joined data source is consistent.
Sets the anchor to the selected data source. The data source that is the anchor is the basis for any joins. Depending on the type of join, changing the anchor to a different data source can change the joined data source fields.

- **Set Access Context**
  Sets the access context to the access context of the selected data source.

**Result**
The joined data source is available for reporting. You can create reports using data from the joined data source. For information about creating reports, see Create a Report [page 341].

**See Also**
Analytics Quick Guide [page 323]

### 6.5.3.4 Create a Report

**Overview**
In the SDK you can create reports by using data sources that you have created. You can then define the report and the report layout in the SAP solution by using the Analytics tools.

You can open a report that you created in the SDK in the Design Reports view in the development environment with the Developer user role. You cannot open reports in the development environment with the Business User role.

**Prerequisites**
- You have created and opened a solution in the SDK.
- You have created and activated a data source. For more information, see:
  - Create a Data Source [page 336]
  - Create a Combined Data Source [page 338]
  - Create a Joined Data Source [page 339]
- You have enabled the key user mode so that the system saves the analytics content you create in your solution. To do this, in the Solution Explorer, right-click your solution and select Enable Key User.
  
  **Tip:** You do not need to enable the key user mode if you are working with a sandbox solution.

- You are assigned the Developer user role and have access rights for the Business Analytics work center. For more information, see User Setup Quick Guide [page 39].

**Procedure**
1. In the Solution Explorer, right-click your data source and select Create Report.
   The logon screen of the SAP solution opens in your browser.
2. Enter the user name and password that you entered when logging on to the repository.
   The Report Wizard opens.
3. Use the Report Wizard to define the report characteristics, key figures, variables, and properties; then define the report layout.

For more information, see Working with Reports Using the Report Wizard.

If you are working with a global solution, the system displays the name of your solution. The report ID starts with the default namespace.

If you are working with a sandbox solution, the system does not display the name of your solution or the report ID.

The system displays the name of the data source you created; however, you can also select another data source.

Result

You have created a report in your solution and defined the report content using the data defined in your data source. You can check which reports are defined for a particular data source. For more information, see Analytics Quick Guide [page 323] under Manage Reports.

You can create a work center and a work center view to which you want your report to be assigned; then, in the UI designer, you can:

1. Add the work center view to the work center.
2. Assign a stable anchor to the work center view to which you want your report to be assigned.
   For more information, see Define an Anchor for a Floorplan [page 560] and Anchors [page 564]. You must assign this anchor so that key users in the SAP solution can assign reports to this work center view.
3. Add the List and Gallery standard reports list views to your work center. These views list all reports that are assigned to the views of the work center.
   For more information, see Add a Report List View to a Work Center [page 559].

Only key users working in a customer’s production system can assign a report that you have created to a work center view. To assign a report to a work center view, the solution that contains the report must therefore be implemented in the customer’s production system.

We recommend that you provide information about the work center views to which customers need to assign reports in your Solution Documentation. For more information, see Documentation Types in the SAP Solution [page 609].

6.6 Extensibility

6.6.1 Business Object Extensions

6.6.1.1 Business Object Extensions Quick Guide

Business object extensions enable you to enhance business objects that are provided by SAP by adding extension fields to them.

After you create and activate a business object extension, you can add the extension field to the corresponding forms, screens, reports, and search categories.
Business and Technical Background

Business Object Extensions

You can customize the business objects that are provided by SAP by creating business object extensions. When you create a business object extension, you modify the extendable nodes of an existing business object to include additional fields. Once the business object extension has been activated, you can include the extension fields in script files, forms, screens, reports, and search categories. Creating a business object extension does not create a copy of the business object. Once you extend a business object, the original business object is no longer available and is replaced by the extended business object.

Most business objects provided by SAP contain both extendable and non-extendable nodes. However, some business objects do not contain any extendable nodes and cannot be used with business object extensions. You can only create business object extensions for business objects that are provided by SAP. Other business objects cannot be used with business object extensions.

Key User Tool

To learn about the tools that customers can also use to create extension fields for a screen that has been enabled for extension fields, see Extension Fields Quick Guide. For an overview of business contexts that have been enabled to support extension fields, see Business Contexts Enabled for Extension Fields.

For example, a key user can add an extension field with the field type “Text” to the business context Account - General Information and then add the extension field to the end-to-end sales process flow. For more information, see Example: Adding an Extension Field to the Sales Process.

Enterprise Search in the SAP Solution

The Enterprise Search capability in the SAP solution allows customers to search the entire system efficiently. For information about how customers use Enterprise Search, see SAP Business ByDesign Enterprise Search.

Tasks

Create a Business Object Extension

You can create a business object extension and add new fields to the extendable nodes. The business object’s extendable nodes are automatically added to a template, which you can modify with extension fields.

For more information about this task, see here [page 345].

Define the Business Logic for a Business Object Extension

You use event script files and validation script files to define the business logic for your business object extension.

The events currently supported are AfterModify and BeforeSave.

The only validation currently supported is onSave.

For more information about this task, see here [page 347].

Add an Extension Field to a Screen

You can add an extension field to a business object’s corresponding screens.

For more information about this task, see here [page 348].
Add an Extension Field to a Form

You can add an extension field to the business object’s corresponding forms. You can either add the extension field to an original form, or you can create a copy of a form and add the extension field to the copy.

For more information about this task, see here [page 348].

Add an Extension Field to a Report

You can add an extension field to a data source that is based on the same business object as the extension field. You can then add the field to any reports based on the data source.

For more information about this task, see here [page 350].

Add an Extension Field to the Enterprise Search

You can add an extension field to an enterprise search category. This allows users to search for the contents of an extension field within a particular category, such as sales orders or suppliers.

For more information about this task, see here [page 352].

Define a Message for a Business Object Extension

You can define a message in a business object extension and raise the message in an event. Messages can only be raised in the business object extension where they are defined.

For more information about this task, see here [page 353].

Add a Label and Tooltip to an Extension Field

You can define a label and a tooltip for an extension field. If you then add the extension field to a screen:

- the label is displayed on the screen instead of the extension field name
- the tooltip is displayed when you place the cursor on the extension field

For more information about this task, see here [page 351].

See Also

Process Extension Scenario Quick Guide  [page 355]
Syntax for Business Object Extension Definitions  [page 170]
Maintenance of Business Object Extensions  [page 344]

6.6.1.2 Maintenance of Business Object Extensions

Overview

A solution is in maintenance mode if it has been assembled and downloaded, that is, the solution status is Assembled. You can make changes to the solution in a patch; however, you can only make restricted changes to certain content types. These change and delete restrictions ensure that you do not make changes to a solution that could lead to loss of data or create inconsistencies or errors on a customer’s production tenant.
Change and Delete Restrictions

The following table gives an overview of the change and delete restrictions for business object extensions in a solution that is assembled. The system performs the change and delete restrictions checks when you save the business object extension.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Adding Allowed</th>
<th>Deletion Allowed</th>
<th>Renaming Allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business object extension</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Business object node element</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Data type of element</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Label annotation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Tooltip annotation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Transient annotation</td>
<td>No</td>
<td>No</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Reference annotation</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Relation annotation</td>
<td>Yes</td>
<td>No</td>
<td>—</td>
<td>The target must not be changed.</td>
</tr>
<tr>
<td>Decimal annotation</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Scenario annotation</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Messages</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Process extension scenario</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Providing default values</td>
<td>Yes</td>
<td>Yes</td>
<td>—</td>
<td>Changes are allowed.</td>
</tr>
</tbody>
</table>

See Also

Business Object Extensions Quick Guide  [page 342]
Lifecycle Management of Customer-Specific Solutions  [page 127]

6.6.1.3 Tasks

6.6.1.3.1 Create a Business Object Extension

Overview

You can create a business object extension and add new fields to the extendable nodes. The business object’s extendable nodes are automatically added to a template, which you can modify with extension fields.

Prerequisites

You know how to work with business objects and how to use the scripting language.
Procedure

1. In the Solution Explorer, expand your solution, right-click your project file, and select Add New Item. The Add New Item dialog appears.
2. Select Business Object Extension.
3. Enter a Name for your business object extension and click Add.

You cannot use a name that is already used for a business object or business object extension in your solution.

The Business Object Selection dialog appears.
4. Select the Namespace of the business object that you want to extend.
5. Select the Business Object that you want to extend.
6. Click OK.

A business object extension item with the file extension .xbo is added to your project. This file contains a template for the business object extension, including the extendable nodes.
7. In the Solution Explorer, double-click the business object extension to open the file in the editing pane.
8. To add extension fields, add elements to the extendable nodes with the scripting language.
9. Click the Save button.
10. Right-click the business object extension and select Activate.

Example

This example shows an extension of the SalesOrder business object. Two text elements, Test_Text1 and Test_Text2, have been added to the business object extension – one at the root level and one at the item level.

import AP.Common.GDT;
import AP.CRM.Global;
[Extension] businessobject AP.CRM.Global:SalesOrder {
    element Test_Text1:LANGUAGEINDEPENDENT_EXTENDED_Text;
    node Item {
        element Test_Text2:LANGUAGEINDEPENDENT_EXTENDED_Text;
    }
}

See Also

Syntax for Business Object Extension Definitions [page 170]
Business Object Extensions Quick Guide [page 342]
Scripting Language [page 156]
Business Objects Quick Guide [page 297]
6.6.1.3.2 Define the Business Logic for a Business Object Extension

Overview
You use event script files and validation script files to define the business logic for your business object extension.

- The events currently supported are AfterModify and BeforeSave.
- The only validation currently supported is OnSave.

Prerequisites
The extended business object must be activated if you want to access an extension field in your script file.

Procedure
1. In the Solution Explorer, right-click your business object extension and select Create Script Files.
   The Create Script Files dialog box is displayed, listing all the nodes of the business object. You can only select nodes that are released as part of the Public Solution Model (PSM).
   To find out which nodes have been released for the PSM, select the Repository Explorer in the SDK. For more information, see Repository Explorer [page 53]
2. Select the script files that you want to create for each of the nodes of the business object.
   Events, such as BeforeSave and AfterModify, can be created for each node. The validation OnSave can only be created for the root node.
3. Select the Mass Enable checkbox if you want the script file to be mass enabled.
   If mass enabled, the script file will be executed for multiple instances of the business object node. You cannot deselect this checkbox after a script file has been created. If you want to modify a script file afterwards so that it will only be executed for a single instance of the business object node, you need to delete the script file first and re-create it again.
   - If you use the this keyword in the coding of a mass enabled script file, this represents a collection of business object node references.
4. Click OK.
   In the Solution Explorer, script files with the file extension .absl appear below the corresponding nodes of your business object. Each script file is opened on a tab page in the code editor.
5. In the code editor, select the script file you want to edit and enter the coding for the event.
   For more information about the methods and keywords you can use, see Syntax for Implementation of Business Logic [page 178].
6. In the File menu, click Save.
7. In the Solution Explorer, right-click the script file and select Activate.

See Also
Actions, Events, and Validations [page 301]
6.6.1.3.3 Add an Extension Field to a Screen

**Overview**

You can add an extension field to a business object’s corresponding screens.

**Prerequisites**

You have created and activated a business object extension that contains at least one extension field.

**Procedure**

1. In the *Solution Explorer*, right-click the business object extension (.xbo file) and select *Enhance Screen*.
   The *Enhance Screen* dialog appears and lists the screens that can be modified for this business object extension.
2. Select the screen to add the extension field to, and click *OK*.
   UI Designer opens.
3. In the *Extensibility Explorer*, select the *Section Group* or *List* to which you want to add the extension field.
4. Depending on the type of entity selected, click either the *Add Extension Field to Section Group* button or the *Add Extension Field to List* button.
   The *Add Extension Field* dialog appears and lists the available extension fields.
5. Select the extension fields to add to the screen by clicking the *Add* check box.
   - If you want the field to be read-only, select the *Display Only* check box.
6. Click *Apply*.
7. Click the *Save* button.
8. Click the *Activate* button.
   The *Activate Worklist* dialog appears.
9. Select the screen you have modified and click *OK*.

**See Also**

Create a Business Object Extension  [page 345]

6.6.1.3.4 Add an Extension Field to a Form

**Overview**

You can add an extension field to the business object’s corresponding forms. You can either add the extension field to an original form, or you can create a copy of a form and add the extension field to the copy.
Prerequisites

- The business object extension has been activated.
- You are working with a global solution (not a sandbox solution), which allows you to enable the key user.

Procedure

1. **Enable the Key User**
   
   In the Solution Explorer, right-click on your solution and select **Enable Key User**.

2. **Create a Copy of a Form (optional)**
   
   You can create a copy of a form and add the extension fields to the copy.
   
   - This option is useful if you want to be able to print different versions of a form. Example: Depending on the selection by the business user, the preview is to show the standard print form or the extended print form.
   
     a. Right-click your business object extension (.xbo file) and click **Enhance Form**.
        
        The **Enhance Form** dialog appears.
      
     b. Click **Copy Form**.
        
        The **Form Template Maintenance** screen appears.
      
        If you are not already logged on to the SAP solution, you will be prompted to log on.
      
     c. In the **Show** list, select **Active Forms** and click **Go**.
        
        The available forms are displayed.
      
     d. Select the form template that you want to copy, and click **Copy As New Template**.
      
     e. Enter a **Template Name** and **Template Description** for your new template.
      
     f. Click **OK**.
      
     g. Click the **Refresh** button to verify that your template has been added to the list of available templates.

   - Use a filter to find your template by name.

3. **Edit the Form in the Easy Form Editor**

   a. Right-click on your business object extension (.xbo file) and click **Enhance Form**.
      
      The **Enhance Form** dialog appears.
    
   b. Select the extension field you want to add to the form and click **OK**.
      
      The **Further Usage of Extension Field** screen appears.
    
      If you are not already logged on to the SAP solution, you will be prompted to log on.
    
   c. Select the form template to which you would like to add the extension field and click **Add Field and Edit Easy Form Editor**.
      
      The **Easy Form Editor** screen appears.
    
   d. In the **Sections** area, select the section to which you would like to add an extension field.
      
      The **Details** area lists the fields that can be added to the section.
e. In the Details area, select the extension field to add to the form, and set the Show attribute to Show always.

f. To preview the form with the extension field, in the Preview area, click the Refresh button.

g. To save your changes, click Save.

h. To make your change visible to other users, click Publish and Close.

Additional Activities
To view the form template in the SAP solution, business users must create a form template rule.

See Also
Create a Business Object Extension  [page 345]
Form Template Selection Quick Guide

6.6.1.3.5 Add an Extension Field to a Report

Overview
You can add an extension field to a data source that is based on the same business object as the extension field. You can then add the field to any reports based on the data source.

The following table shows whether an extension field can be used as a key figure or characteristic.

<table>
<thead>
<tr>
<th>Extension Field Type</th>
<th>Analytics Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount</td>
<td>Key Figure</td>
</tr>
<tr>
<td>Date</td>
<td>Characteristic</td>
</tr>
<tr>
<td>Decimal Number</td>
<td>Key Figure</td>
</tr>
<tr>
<td>E-Mail Address</td>
<td>Characteristic</td>
</tr>
<tr>
<td>Indicator</td>
<td>Characteristic</td>
</tr>
<tr>
<td>List</td>
<td>Characteristic</td>
</tr>
<tr>
<td>Text</td>
<td>Characteristic</td>
</tr>
<tr>
<td>Time</td>
<td>Characteristic</td>
</tr>
<tr>
<td>Quantity</td>
<td>Key Figure</td>
</tr>
<tr>
<td>Web Address</td>
<td>Characteristic</td>
</tr>
</tbody>
</table>

Prerequisites
- The business object extension has been activated.
- You are working with a global solution (not a sandbox solution), which allows you to enable the key user.

Procedure
1. In the Solution Explorer, right-click on your solution and select Enable Key User.
2. Right-click on your business object extension (.xbo file) and click Enhance Report. The Enhance Report dialog box appears.
3. Select the extension field you want to add to the report and click OK. The Further Usage of Extension Field screen appears.

If you are not already logged on to the SAP solution, you will be prompted to log on.

4. On the Data Sources and Reports tab, you can view all data sources available for the business context in which you created the field. When you select a data source, all reports based on that data source are displayed in the lower table.
5. To add the field to a data source, select the data source and click Add Field.
6. To add the field to an existing report based on this data source, select the report from the lower table and click Add Field.

The report opens in key user mode.
7. In the Add Fields dialog box, select the Display in Report checkbox next to the extension field. Characteristics are added to the report under Not Currently Shown and key figures are available under Columns.
8. Click OK and close the report without saving.

The field will be available to all business users when they next access the report.

See Also

Create a Business Object Extension [page 345]

6.6.1.3.6 Add a Label andTooltip to an Extension Field

Overview

You can define a label and a tooltip for an extension field. If you then add the extension field to a screen:
• the label is displayed on the screen instead of the extension field name
• the tooltip is displayed when you place the cursor on the extension field

Procedure

1. Create an extension field with a label and a tooltip using the following syntax:
   [Label ("<label name>")] [Tooltip ("<tooltip name>")] element <element name> : <data type>;
   For more information, see Create a Business Object Extension [page 345].
2. Add the extension field to a screen. For more information, see here [page 348].

Result

In the Solution Explorer in the SDK, right-click your extended screen (.xuicomponent) and select Preview. The label is displayed for the extension field. The tooltip is displayed when you place the cursor on the field.
6.6.1.3.7 Add an Extension Field to the Enterprise Search

Overview
You can add an extension field to an enterprise search category. This allows users to search for the contents of an extension field within a particular category, such as sales orders or suppliers.

The following table shows which type of search can be used for each extension field type:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Basic Worklist Search</th>
<th>Advanced Worklist Search</th>
<th>Basic Enterprise Search</th>
<th>Advanced Enterprise Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Date</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Decimal Number</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>E-Mail Address</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Indicator</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>List</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Text</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Time</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Quantity</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Web Address</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Prerequisites
- The business object extension has been activated.
- You are working with a global solution.

Procedure
1. In the Solution Explorer, right-click on your solution and select Enable Key User.
2. Right-click on your business object extension (.xbo file) and click Enhance Enterprise Search. The Enhance Enterprise Search dialog box appears.
3. Select the extension field you want to add to the search and click OK. The Further Usage of Extension Field screen appears.
4. If you are not already logged on to the SAP solution, you will be prompted to log on.
5. On the Enterprise Search tab, you can view all enterprise search categories to which you can add the field. This is determined by the business context in which you created the field.
6. To add a field to an enterprise search category, select it from the list and click Add Field.

The enterprise search category determines in which category users will be able to search for the contents of this field using the basic enterprise search. They can then refine their search by clicking on Advanced and using the extension field as a search parameter.
See Also
SAP Business ByDesign Enterprise Search

6.6.1.3.8 Define a Message for a Business Object Extension

Overview
You can define a message in a business object extension and raise the message in an event. Messages can only be raised in the business object extension where they are defined.
The following message types are available:
- Information
- Warning
- Error
- Success

Prerequisites
- You have an open solution in the SDK.
- You have created a business object extension (.xbo) as part of your solution. This extension contains at least one element and is activated.

Procedure
1. In the Solution Explorer, double-click the business object extension to open the file in the editing pane.
2. Use the message and text keywords to specify the name and content of the message, respectively.
3. Use the raises keyword to specify which messages are raised for the individual nodes of the business object extension.
import AP.Common.GDT;
import AP.CRM.Global;

[Extension] businessobject AP.CRM.Global:Opportunity raises
Information_Message, Error_Message, Warning_Message {
  // You must activate this business object before you can access the
  // extension fields
  // in script files, forms, and screens.
  message Information_Message text "This is an Information message";
  message Error_Message text "This is an Error message with one parameter
  &1" : LANGUAGEINDEPENDENT_SHORT_Name;
  message Warning_Message text "This is a Warning message with two
  parameters &1 and &2" : LANGUAGEINDEPENDENT_SHORT_Name, IntegerValue;
  message Success_Message text "This is a Success message with a variable
  parameter &1" : Amount;
  element Test_Extension_Field : Amount;
  node Item raises Information_Message, Success_Message {
  }
}

You can include up to four parameters. For each parameter, you must specify the data type.

4. In your action or event script file, raise the message using the raise keyword.

import ABSL;
raise Information_Message.Create("I");
raise Error_Message.Create("E", "Error");
raise Warning_Message.Create("W", "Warning", 25);
raise Success_Message.Create("S", this.Test_Extension_Field);

Code completion is available after you type the raise keyword.

Result
Messages appear on the user interface based on the actions or events that you defined.

See Also
Scripting Language  [page 156]
Business Object Extensions Quick Guide  [page 342]
6.6.2 Process Extension Scenarios

6.6.2.1 Process Extension Scenario Quick Guide

A process extension scenario links the data from one business context to other related business contexts. You create a process extension scenario from a list of predefined extension scenarios. For example, you can create a process extension scenario that links the following business contexts:

- **Lead - General Information > Opportunity - General Information**
- **Opportunity - General Information > Sales - General Information**

Each process extension scenario contains one or more data flows. Each data flow consists of a source and target business context. This reflects how the data is passed from one business context to the next as part of a business process.

- **Opportunity - General Information > Sales - General Information**
  - Opportunity - General Information > Sales Quote - General Information
  - Opportunity - General Information > Sales Order - General Information

For a list of all predefined extension scenarios, see Extension Scenarios [page 357].

If you now extend any of the business objects involved in the scenario, you can use the annotation [Scenario] to declare that the element is available to all the business contexts specified in the scenario.

For example, if you add an element to the Lead business object, you can use an extension scenario to make this element available if you extend the Opportunity, Sales Quote, or Sales Order. When an Opportunity is created from the Lead, the data for your extension field will be passed to the Opportunity. Similarly, when a Sales Order is created from the Opportunity, the data will be passed to the Sales Order.

**B2B Message Extension**

You can also extend B2B messages in a process extension scenario. If a predefined extension scenario includes an inbound or outbound B2B data flow, you can create a process extension scenario that includes this data flow. Once an extension field is associated to a scenario that includes such a flow, the extension field value will be propagated to the B2B message during runtime.

In the SDK Feature Pack 3.5, the only B2B messages that can be extended are the 3PL (third-party logistics) messages.

When selecting the data flows to include in a process extension scenario, you can identify outbound and inbound B2B messages by their individual icons.
Tasks

Create a Process Extension Scenario
For more information about creating a process extension scenario, see here  [page 358].

Add an Extension Field to a Process Extension Scenario
For more information about adding an extension field to a process extension scenario, see here  [page 359].

Example: Sales Tracking
For an example of adding an extension field to a process extension scenario, see here  [page 360].
### 6.6.2.2 Extension Scenarios

**Overview**

Extension scenarios enable you to create an extension field in one business context and then add that extension field to multiple other business contexts. The data stored in the extension field is then passed from one business context to the next as part of a business process.

The following table contains a list of extension scenarios predefined by SAP.

<table>
<thead>
<tr>
<th>Extension Scenarios</th>
<th>Project - General Information to Snapshot - General Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account - General Information to Lead - General Information</td>
<td>Project - Task to Baseline - Task</td>
</tr>
<tr>
<td>Account - General Information to Opportunity - General Information</td>
<td>Project - Task to Snapshot - Task</td>
</tr>
<tr>
<td>Account - General Information to Sales - General Information</td>
<td>Project - Team Member to Baseline - Team Member</td>
</tr>
<tr>
<td>Account - General Information to Service - General Information</td>
<td>Project - Team Member to Snapshot - Team Member</td>
</tr>
<tr>
<td>Activities</td>
<td>Project - Work Package to Baseline - Work Package</td>
</tr>
<tr>
<td>Business Partner - General Information to File for Account Export Run</td>
<td>Project - Work Package to Snapshot - Work Package</td>
</tr>
<tr>
<td>Customer Invoice - Item to Customer Return - Item</td>
<td>Purchase Request - Item to Purchase Order - Item</td>
</tr>
<tr>
<td>Customer Return - Item to Invoice Requests - Item</td>
<td>Sales - General Information</td>
</tr>
<tr>
<td>Inbound Delivery - General Information to Outbound Delivery - General Information</td>
<td>Sales - General Information</td>
</tr>
<tr>
<td>Inbound Delivery Notification - General Information to Inbound Delivery - General Information</td>
<td>Sales - General Information to Service - General Information</td>
</tr>
<tr>
<td>Inbound Delivery Notification - Item to Inbound Delivery - Item</td>
<td>Sales - General Information</td>
</tr>
<tr>
<td>Inspection Plan - General Information to Inspection - Snapshot</td>
<td>Sales - Item</td>
</tr>
<tr>
<td>Invoice Requests - General Information to Customer Invoice - General Information</td>
<td>Sales Order - General Information to Invoice Requests - General Information</td>
</tr>
<tr>
<td>Invoice Requests - General Information to Customer Invoice - Item</td>
<td>Sales Order - Item to Invoice Requests - Item</td>
</tr>
<tr>
<td>Invoice Requests - Item to Customer Invoice - Item</td>
<td>Sales Order - Item to Delivery Request - Item</td>
</tr>
<tr>
<td>Lead - General Information to Opportunity - General Information</td>
<td>Sales Order - Item to Purchase Request - Item</td>
</tr>
<tr>
<td>Lead - General Information to Sales - General Information</td>
<td>Service - Affected Product</td>
</tr>
<tr>
<td>Opportunity - General Information to Sales - General Information</td>
<td>Service - General Information</td>
</tr>
<tr>
<td>Opportunity - Item to Sales - Item</td>
<td>Service - General Information to Sales - General Information</td>
</tr>
<tr>
<td>Outbound Delivery - Item to Invoice Requests - Item</td>
<td>Service - General Information to Sales - General Information</td>
</tr>
<tr>
<td>Outbound Delivery Request - Item to Outbound Delivery - Item</td>
<td>Service Confirmation - General Information to Invoice Requests - General Information</td>
</tr>
<tr>
<td>Product - General Information to Goods and Activity Confirmation - Product Item</td>
<td>Service Confirmation - Item to Invoice Requests - Item</td>
</tr>
</tbody>
</table>
6.6.2.3 Tasks

6.6.2.3.1 Create a Process Extension Scenario

Overview

You can create a process extension scenario to link the data from one business context to other related business contexts.

Prerequisites

You have an open solution in the SDK.

Procedure

1. In the **Solution Explorer**, right-click your project and select **Add New Item**. The **Add New Item** dialog box is displayed.
2. Under **Installed Templates**, click **SAP Extension**.
3. Select the **Process Extension Scenario** template, enter a **Name** for the process extension scenario, then click **Add**. The **Extension Scenario Creation** dialog box is displayed.
4. Select the **Namespace**, **Business Object**, and **Node**. Based on your selection, a list of possible extension scenarios is displayed. The individual data flows are displayed under each extension scenario.
5. Select one or more extension scenarios. Based on your selection, additional extension scenarios may be displayed.
6. Select additional extension scenarios, as desired.

7. Click OK.

A process extension scenario item with the file extension .xs is added to your project.

See Also
Add an Extension Field to a Process Extension Scenario [page 359]
Example: Sales Tracking [page 360]

6.6.2.3.2 Add an Extension Field to a Process Extension Scenario

Overview
You can use the annotation [Scenario] to declare that an element should be available to all the business contexts specified in a process extension scenario.

For example, if you have a process extension scenario that links the Lead, Opportunity and Sales Order business contexts, you can add an element to the Lead business object, and this element will be available if you extended a screen for an Opportunity, a Sales Quote, or a Sales Order.

Prerequisites
- You have an open solution in the SDK.
- You have created a process extension scenario as part of your solution.

Procedure
1. In the Solution Explorer, right-click your project and select Add New Item. The Add New Item dialog box is displayed.
2. Under Installed Templates, click SAP Extension.
3. Select the Business Object Extension template, enter a Name for your business object extension, then click Add. The Business Object Selection dialog box is displayed.
4. Select the Namespace of the business object that you want to extend.
5. Select the Business Object that you want to extend.
6. Click OK.

A business object extension item with the file extension .xbo is added to your project. This file contains a template for the business object extension, including the extendable nodes.

7. In the Solution Explorer, double-click the business object extension to open the file in the editing pane.
8. To add extension fields, add elements to the extendable nodes with the scripting language.
9. To include an element in an extension scenario, use the following syntax:

   [Scenario (Lead_to_Sales)] element ext_field_Sales_root:Text;
Process extension scenarios are specific to the node for which they were created. If an extension scenario was created for the root node, then it can only be called at the root node within the .xbo. Similarly, a process extension scenario created for the item node can only be called from the item node within the .xbo.

10. Click the Save button.
11. Right-click the business object extension and select Activate.

See Also
Scripting Language [page 156]
Business Object Extensions Quick Guide [page 342]
Create a Process Extension Scenario [page 358]
Example: Sales Tracking [page 360]

6.6.2.4 Example: Sales Tracking

Overview
Your company generates sales from a number of sources, such as print advertisements, promotional campaigns, and mailing lists. You want to track the source of your sales from the time the lead is generated until a sales order is created. To achieve this, you will perform the following steps:

- Extend the Lead business object to include a new field called “Sales_Lead_Source”.
- Create a process extension scenario that links the business contexts for Lead, Opportunity and Sales.
- Use the [Scenario] annotation to make your extension field available to all the business contexts in the process extension scenario. When an Opportunity is created from the Lead, the data for your extension field will be passed to the Opportunity. Similarly, when a Sales Order is created from the Opportunity, the data will be passed to the Sales Order.
- Enhance the Lead quick activity floorplan (QAF) to include the “Sales_Lead_Source” extension field. This allows you to enter data for this field when creating a Lead.
- Enhance the Sales Order QAF to display the “Sales_Lead_Source” field.

You could also extend any of the floorplans for either the Opportunity or the Sales Quote with the same extension field.

Prerequisites

- You are familiar with the SAP solution.
- You are familiar with the following processes within the SDK:
  - Create a Process Extension Scenario [page 358]
  - Create a Business Object Extension [page 345]
  - Add an Extension Field to a Screen [page 348]
- You have an open solution in the SDK.
Procedure

1. Create an extension scenario for the Root node of the Lead business object.
   Give the extension scenario the name Sales_Lead_Tracking and select the following business contexts:
   - Lead - General Information > Opportunity - General Information
   - Opportunity - General Information > Sales - General Information

2. Create an extension for the Lead business object and give it the name Lead_BO_Extension.
   a. Add a text element called Sales_Lead_Source to the root node.
   b. Add an annotation for the Sales_Lead_Tracking process extension scenario.

   ```java
   import AP.Common.GDT;
   import AP.CRM.Global;
   [Extension] businessobject AP.CRM.Global:Lead {

   // You must activate this business object before you can access the extension fields
   // in script files, forms, and screens.

   [Scenario (Sales_Lead_Tracking)] element Sales_Lead_Source : Text;
   }
   ```
As an additional step, you can specify a Label and Tooltip for this element:

```java
[Label("Sales_Lead_Label") [Tooltip("Sales_Lead_Tooltip")]
``` 

3. **Enhance the QAF for the Lead to include the extension field in the Classification section group.**

4. **Create an extension for the Sales Order business object and give it the name Sales_Order_BO_Extension.**

   You do not need to add any elements to this extended business object, as it is included in the extension scenario and can therefore access the extension field you added to the Lead business object.

5. **Enhance the QAF for the Sales Order to include the extension field in the General section group.**

6. **Log on to the SAP solution with a user that has access to the New Business, Marketing, and Sales Orders work centers.**

7. **In the Marketing work center, create a new Lead.** Notice that your extension field is displayed on the QAF.
8. Enter the text **Online Advertising** in the extension field, then complete all required fields and save the Lead.

9. Create an Opportunity from the Lead.

10. Create a Sales Order from the Opportunity.
    
    Notice that your extension field is displayed on the QAF and is populated with the text you entered into the Lead QAF.
6.6.3 Enhancement Implementations

6.6.3.1 Enhancement Implementations Quick Guide

SAP provides enhancement options based on SAP technology that allow you to enhance the standard behavior of specific applications of the SAP solution without modifying the standard solution. In the SDK, you can create enhancement implementations of SAP enhancement options. For example, you can implement a custom variant of a calculation or override the default filter values to meet country-specific, industry-specific, or customer-specific requirements. You can define your own business logic, for example, for determining, validating or mapping data.

Business and Technical Background

SAP Enhancement Options

An enhancement option is a repository object provided by SAP for a specific application to allow the standard behavior of the application to be enhanced without modifying the standard solution. Enhancement options are defined with metadata and documentation. A distinction is made between single-use enhancement options and multiple-use enhancement options. If an enhancement option is for single use, it can be implemented only once in a solution. If an enhancement option is for multiple use, it can be implemented more than once in a solution. The documentation of the enhancement options describes how you can use an enhancement option, when the enhancement option is called, and what it does. Furthermore, it gives information about the business configuration.
content you need to develop for an enhancement implementation, the scoping requirements in the customer system, as well as information about the filters, the input parameters and output parameters, and testing activities. You can explore SAP enhancement options and view the documentation of the enhancement options using the Repository Explorer tool window. For more information, see Repository Explorer [page 53].

Use Case for SAP Enhancement Options

Creation of Bank Statement File Formats

You can create bank statement formats to meet the business requirements of customers located in or dealing with countries for which a localized version of the SAP Business ByDesign solution is not available. You can then map the a bank statement file to the structure of the Bank Statement Format business object. The customer can then update the system scoping and use this format when processing automatic bank statements.

For information about the end-to-end process, see Creating Bank Statement File Formats.

Tasks

Create a Business Configuration Set

You can create a business configuration set (BC set) using an SAP business configuration object (BCO) to configure the behavior of standard business processes in the SAP solution. If you create an implementation of an enhancement option which is for single use only, you can also define a BC set using an SAP BCO so that you can enhance the filter values.

For customer-specific solutions:

- You can only create a BC set using an SAP BCO if you are developing the solution on a development tenant. For information, see Lifecycle Management of Customer-Specific Solutions on a Development Tenant [page 139].
- You cannot create BC sets using SAP BCOs in a solution template.

For more information, see here [page 290].

Create an Enhancement Implementation

You can create an enhancement implementation of an SAP enhancement option to allow customers to apply business rules that meet specific requirements, for example, country-specific requirements, without modifying the standard functionality of the SAP solution. An enhancement implementation consists of a message definition, a filter, and a script file.

For more information, see here [page 366].

Test an Enhancement Implementation

To check content in the business adaptation catalog, for example, for testing purposes, you must be assigned the Business User role and have access rights for the Business Configuration work center. For more information, see User Setup Quick Guide [page 39].

When you use the Deploy Business Configuration function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the Deploy Business Configuration function deploys all business configuration content independently of any business options you have created.
6.6.3.2 Create an Enhancement Implementation

Overview

You can create an enhancement implementation of an SAP enhancement option to allow customers to apply business rules that meet specific requirements, for example, country-specific requirements, without modifying the standard functionality of the SAP solution. An enhancement implementation consists of a message definition, a filter, and a script file.

Prerequisites

- You have created and opened a solution in the SDK.
- If the enhancement option for which you created an implementation is for single use only, you have defined a business configuration set (BC set) using an SAP business configuration object (BCO) so that you can enhance the filter with the value that you want to use in your filter definition. You must also assign the BC set to a business option.

For information about which SAP BCO is related to which filter, see the public solution model (PSM) documentation for the enhancement option. For information about creating a BC set, see Create a BC Set Using an SAP BCO [page 290].

Procedure

1. In the Solution Explorer, click the Add New Item button.
   The Add New Item dialog box opens.
2. In the Add New Item dialog box, select the Enhancement Implementation template, enter a name for your enhancement implementation, and then click Add.
   The Create Enhancement Implementation dialog box opens.

   You must specify a name that begins with an uppercase letter (A-Z) and only contains alphanumeric characters (A-Z, a-z, 0–9) and underscore characters (_). The name must not exceed 120 characters.

3. In the Create Enhancement Implementation dialog box, select a namespace and an enhancement option.
   Optionally, you can select a namespace and business object to filter the list of enhancement options.
   A new item with the file extension .enht appears in the Solution Explorer. By default, the following items are created under the .enht file:
   - Filter
     An item with the same name as the enhancement implementation and the file extension .fltr
   - Script file
     An item with the file extension .absl, for example, CONVERT.absl

   The .enht file opens in a document window.
4. Under Message Definition, define messages for your enhancement implementation by doing the following:
   a. Specify a message ID.
      Each message you define must have a unique ID.
b. Define the messages parameters one-by-one, from left to right, by doing the following:

1. In the parameter cell, double-click [ ].
   The Parameter Selection dialog box opens. By default, the http://sap.com/xi/AP/Common/GDT is displayed and the Data Type field is blank.
2. Select a namespace and a data type.

c. Specify one or more message texts.
   You use ampersands (&) in the message text to represent placeholders. The number of placeholders in the text must match the number of parameters. For information about defining message texts, see the Business Center at https://www.sme.sap.com under SAP Business ByDesign Community Wiki SAP Business ByDesign Studio Documentation in the section Documentation Style Guides Style Guide for Message Texts.

d. Save your changes; then, in the Solution Explorer, right-click the .enht file and select Check In Content.
   The item is locked for editing.

5. In the Solution Explorer, double-click the .fltr file.
   The filter editor opens in a document window.

6. Under Filters, specify a value for each filter.
   • By default, all filters are disabled except for mandatory filters.
   • To enable a filter, you must select the filter checkbox.
   • If you do not specify a value for a mandatory filter, the system reads the value as blank.
   • If a code list is available for a filter, you can select a value from a dropdown list.
   • If you hover over the filter name, the data type of the filter is displayed as a tooltip.

7. Save your changes; then, in the Solution Explorer, right-click the .fltr file and select Check In Content.
   The item is locked for editing.

8. In the Solution Explorer, double-click the .absl file.
   The script file opens in a document window. By default, the following code is displayed:
   ```
   import ABSL;
   var result : AEString;
   return result;
   ```

9. In the code editor, define the business logic for your enhancement implementation. For information about defining business logic, see Syntax for Implementation of Business Logic [page 178].

10. Save your changes; then, in the Solution Explorer, right-click the .absl file and select Check In Content.
    The item is locked for editing.

11. In the Solution Explorer, right-click the .enht file and select Activate to activate the messages, the filters, and the business logic you defined.

Result
You have created and defined messages, filters, and the business logic for your enhancement implementation. You can now test your enhancement implementation in SAP Business ByDesign.

See Also
Enhancement Implementations Quick Guide [page 364]
6.7 Mashups

6.7.1 Mashups Quick Guide

Using the software development kit (SDK), you can create mashups for the SAP solution, and add them to an SAP floorplan or a floorplan that you have designed.

Business and Technical Background

Creating Mashups

Using the software development kit (SDK), you can create mashups for the SAP solution, and add them to an SAP floorplan or a floorplan that you have designed.

For more information, see here [page 369].

Mashup Categories

When you create a new mashup, you need to select a mashup category. Mashup categories are used to group together mashups by the type of service or information that they provide. Depending on the mashup category selected, you can then select a port binding. This defines the screen out-port parameters that can be used in a mashup and the screens on which the mashup can then be made visible.

For more information, see here [page 371].

Mashups and Web Services

Mashups are used to integrate data from SAP’s on-demand solution with data provided by an online Web service or application. Users can access the content provided by these Web services and applications, and use it in their daily work. Mashups can include Web searches, company or industry business information, or online map searches.

For more information, see here [page 379].

Tasks

Create a Port Binding for a Mashup

For information about this task, see here [page 382].

Create a URL Mashup

For information about this task, see here [page 384].

Create an HTML Mashup

For information about this task, see here [page 389].

Create a Data Mashup

For information about this task, see here [page 386].
Add a Mashup to an SAP Floorplan
For information about this task, see here [page 397].

Add a Mashup to Your Floorplan
For information about this task, see here [page 405].

Save and Activate your Changes
After you have made any of the changes listed above, you need to save and activate the changes.

1. From the File menu, choose Save.
2. From the File menu, choose Activate.
3. In the Activate Worklist dialog box, select the check boxes beside the relevant components and click OK.

6.7.2 Business and Technical Background

6.7.2.1 Creating Mashups

Overview
Mashups are used to integrate data from SAP’s on-demand solution with data provided by an online Web service or application. Users can access the content provided by these Web services and applications, and use it in their daily work. Mashups can include Web searches, company or industry business information, or online map searches.

Using the software development kit (SDK), you can create mashups for the SAP solution, and add them to an SAP floorplan or a floorplan that you have designed.

Prerequisites
The following prerequisites apply:

- Mashups can be created for global solutions only. (Global solutions have names that do not start with “Z”, and are transported and deployed to customers). To create a global solution, ensure that the Create Sandbox Solution flag is not selected.

It does not matter which type of solution you are using. It does not necessarily need to be of the type “mashup”.

- You must enable the key user mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, in the Solution Explorer, right-click the solution and select Enable Key User. For more information about the key user mode, see Solution Explorer [page 56].

If you create a Web service for a mashup, the following prerequisites apply concerning API keys and authentication:

- API Keys
  Some Web services require a unique API key to allow you to access their services. You can generate a company-specific API key on the service provider’s Web site and then use this key to authenticate access to the service from SAP’s on-demand solution.
You can identify which mashups require an API key by the icon displayed in the **API Keys** column. The **Enter API Keys** dialog box displays the API keys that are required by the selected mashup.

For a complete list of all SAP partners and their respective Web sites, see [Web Services from our Partners](#) in the Business Center.

- **Basic Authentication**
  
  Basic authentication allows you to enter a user name and password for authenticating access to a Web service. This user name and password is then used to automatically authenticate all other users accessing the service.

- **Secure Authentication**
  
  For Web services that support the OAuth authentication protocol, you can set up secure authentication. The OAuth protocol enables secure, user-based access to Web services from SAP’s on-demand solution. Unlike other authentication methods, OAuth gives the user full control of his or her own authentication details. To use a mashup based on an OAuth authenticated Web service, users need to authenticate themselves on the service provider’s web site, where they are given a unique PIN code. This PIN code is then used to authenticate access to the service from SAP’s on-demand solution.

**Features**

The following mashup types are available:

There are four different mashup types:

- **URL Mashups**
  
  A mashup that sends data from SAP’s on-demand solution to the URL of an online service provider. The service provider uses the data, for example, to perform a search, and the results are displayed in a new browser window.

- **HTML Mashups**
  
  A mashup that embeds an HTML or JavaScript based Web page directly on a screen.

- **Data Mashups**
  
  A mashup combines and displays data from both internal and external sources. To create a data mashup, an integrated authoring tool is used to transform or merge external Web services with internal business data using industry-standard Web service protocols.

- **Custom Mashups**
  
  A mashup that has been created as an add-on solution by SAP.

You can create new data, URL, and HTML mashups. Custom mashups can be created only by SAP.

In particular, mashups are distinguished by the following characteristics:

- **Mashup Categories**
  
  Mashups are grouped together into categories according to the type of service or information that they provide.

- **Port Binding**
  
  Certain fields on screens are predefined as screen out-ports and in-ports. These screen out-ports and in-ports are grouped together as a port binding. The port binding defines the type of information that can be used in the mashup and defines on which screens you can use the mashup. The mashup category specifies which port bindings are available.

For more information, see [Mashup Categories](#) [page 371].

**Process Flow**

The following process steps apply:

1. **Create a Mashup**
You can create the following mashup types:

- Create a URL Mashup [page 384]
- Create an HTML Mashup [page 389]
- Create a Data Mashup [page 386]

2. **Add the Mashup to a Floorplan**
   After that, you can add the mashup to an SAP floorplan or a floorplan that you have designed. For more information, see:
   - Add a Mashup to an SAP Floorplan [page 397]
   - Add a Mashup to Your Floorplan [page 405]

   If you want to integrate a mashup into a floorplan provided by SAP, this floorplan must have an existing outport with a stable anchor and a referenced anchor. You cannot create your own outports for SAP floorplans.

**See Also**

Mashups Quick Guide [page 368]

### 6.7.2.2 Mashup Categories

**Overview**

When you create a new mashup, you need to select a mashup category. Mashup categories are used to group together mashups by the type of service or information that they provide. Depending on the mashup category selected, you can then select a port binding. This defines the screen out-port parameters that can be used in a mashup and the screens on which the mashup can then be made visible.

**Features**

The following table gives an overview of the different mashup categories available, and the type of information available for use in each mashup category. The out-port type corresponds to one or more parameters that can be used when creating a mashup.
## Mashup Categories and Port Bindings

<table>
<thead>
<tr>
<th>Mashup Category</th>
<th>Port Binding</th>
<th>Description</th>
<th>Out-Port Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business &amp; Finance</strong></td>
<td>Additional Account Information</td>
<td>Show additional information for a dedicated account</td>
<td>Account name information</td>
</tr>
<tr>
<td></td>
<td>Company Financial and Business Information by Company Name or DUNS</td>
<td>Search for business-related information based on a company name or D-U-N-S number</td>
<td>D-U-N-S number and name of company</td>
</tr>
<tr>
<td></td>
<td>Company Financial and Business Information by Company Name</td>
<td>Search for business-related information based on a company name</td>
<td>Name of company</td>
</tr>
<tr>
<td></td>
<td>Company Financial and Business Information by Industry</td>
<td>Search for business-related information based on an industry</td>
<td>Name of industry</td>
</tr>
<tr>
<td></td>
<td>Company Financial and Business Information by Nielsen Code</td>
<td>Search for business-related information based on a Nielsen code</td>
<td>Nielsen code</td>
</tr>
<tr>
<td></td>
<td>Point-of-Sale Transaction Lookup</td>
<td>Look up a point-of-sale transaction in a third-party point-of-sale system</td>
<td>Point-of-sale transaction information</td>
</tr>
<tr>
<td></td>
<td>Web Feed by Company Name</td>
<td>Display web feeds filtered by a company name</td>
<td>Name of company</td>
</tr>
<tr>
<td><strong>Location &amp; Travel</strong></td>
<td>Business Locator</td>
<td>Search for a local business address by address or industry</td>
<td>Industry and address information</td>
</tr>
<tr>
<td></td>
<td>Route Planner</td>
<td>Plan a business route based on a list of addresses</td>
<td>Start address, end address, and additional destinations</td>
</tr>
<tr>
<td></td>
<td>Weather</td>
<td>Display weather information based on an address</td>
<td>Address information</td>
</tr>
<tr>
<td></td>
<td>Web Map</td>
<td>Display a business address on a map or get directions based on an address</td>
<td>Address information</td>
</tr>
<tr>
<td><strong>News &amp; Reference</strong></td>
<td>Business Partner Web Search</td>
<td>Search for a business partner by name</td>
<td>Search term</td>
</tr>
<tr>
<td></td>
<td>Product Web Search</td>
<td>Search for product information by name</td>
<td>Product search term</td>
</tr>
<tr>
<td></td>
<td>Search Provider</td>
<td>Search for a search term</td>
<td>Search term</td>
</tr>
<tr>
<td><strong>Productivity &amp; Tools</strong></td>
<td>Reverse Lookup</td>
<td>Look up a business address using partial address information</td>
<td>Name and address information</td>
</tr>
<tr>
<td></td>
<td>Reverse Lookup by Phone</td>
<td>Look up a business address using a phone number</td>
<td>Phone number</td>
</tr>
<tr>
<td></td>
<td>Shipping Service</td>
<td>Integrate shipping services (limited)</td>
<td>Shipping information</td>
</tr>
<tr>
<td></td>
<td>Sourcing</td>
<td>Search for sourcing information based on a product description</td>
<td>Search term</td>
</tr>
<tr>
<td></td>
<td>URL Navigation</td>
<td>Navigate to an internal or external resource URL</td>
<td>URL navigation information</td>
</tr>
<tr>
<td><strong>Social &amp; Communication</strong></td>
<td>Social Network</td>
<td>Display people or business partner profiles that are listed in a social network</td>
<td>People or business partner profile information</td>
</tr>
</tbody>
</table>
The following table shows on which screens a mashup can be made visible for each port binding.

**Port Binding by Screen**

<table>
<thead>
<tr>
<th>Port Binding</th>
<th>Work Center</th>
<th>Screen</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Account Additional Information</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Account Management</td>
<td>Account Overview fact sheet</td>
</tr>
<tr>
<td><strong>Business Locator</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Account Management</td>
<td>Partner Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Private Account editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contact Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td>Liquidity Management</td>
<td>My Bank Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td>Marketing</td>
<td>Competitor Overview fact sheet</td>
</tr>
<tr>
<td><strong>Supplier Base</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Supplier Base</td>
<td>Suppliers view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supplier editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supplier Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Supplier activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Service Agent Overview fact sheet</td>
</tr>
<tr>
<td><strong>Tax Management</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tax Management</td>
<td>Tax Authority Overview fact sheet</td>
</tr>
<tr>
<td><strong>Business Partner Web Search</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>and Search Provider</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Account Management</td>
<td>Partner Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contact Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contact editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Contact activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Account Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accounts view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corporate Account editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Corporate Account activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Private Account editor</td>
</tr>
<tr>
<td></td>
<td>Business Partner Data</td>
<td>Business Partner Organization editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business Partner Person editor</td>
</tr>
<tr>
<td></td>
<td>Liquidity Management</td>
<td>My Bank Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td>Marketing</td>
<td>Competitor Overview fact sheet</td>
</tr>
<tr>
<td><strong>Personnel Administration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Personnel Administration</td>
<td>New Social Insurance Fund activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Social Insurance Fund Overview fact sheet</td>
</tr>
<tr>
<td><strong>Supplier Base</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Supplier Base</td>
<td>Suppliers view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supplier editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Supplier activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Service Agent Overview fact sheet</td>
</tr>
<tr>
<td><strong>Tax Management</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tax Management</td>
<td>Tax Authority Overview fact sheet</td>
</tr>
<tr>
<td>Port Binding</td>
<td>Work Center</td>
<td>Screen</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td><strong>Company Financial and Business Information by Company Name</strong></td>
<td><strong>Account Management</strong></td>
<td><strong>Account Overview fact sheet</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Accounts view</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Contact editor</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Corporate Account editor</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>New Contact activity</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>New Corporate Account activity</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>New Partner activity</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Private Account editor</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Partner Overview fact sheet</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Partner editor</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Partners view</strong></td>
</tr>
<tr>
<td><strong>Business Partner Data</strong></td>
<td><strong>Business Partner Organization editor</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Business Partners view</strong></td>
</tr>
<tr>
<td><strong>Liquidity Management</strong></td>
<td><strong>Clearing Houses view</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Clearing House editor</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>My Bank Overview fact sheet</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>My Bank editor</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>My Banks view</strong></td>
</tr>
<tr>
<td><strong>Marketing</strong></td>
<td><strong>Competitors view</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Competitor editor</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Competitor Overview fact sheet</strong></td>
</tr>
<tr>
<td><strong>Personnel Administration</strong></td>
<td><strong>Social Insurance Fund editor</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Social Insurance Fund Overview fact sheet</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>New Social Insurance Fund activity</strong></td>
</tr>
<tr>
<td><strong>Supplier Base</strong></td>
<td><strong>Suppliers view</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Supplier editor</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>New Supplier activity</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Supplier Overview fact sheet</strong></td>
</tr>
<tr>
<td><strong>Tax Management</strong></td>
<td><strong>Tax Authority Overview fact sheet</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tax Authority editor</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tax Authorities view</strong></td>
</tr>
<tr>
<td><strong>Company Financial and Business Information by DUNS Number</strong></td>
<td><strong>Account Management</strong></td>
<td><strong>Account Overview fact sheet</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Accounts view</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Corporate Account editor</strong></td>
</tr>
<tr>
<td><strong>Supplier Base</strong></td>
<td><strong>Supplier editor</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Suppliers view</strong></td>
</tr>
<tr>
<td>Port Binding</td>
<td>Work Center</td>
<td>Screen</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Company Financial and Business Information by Industry</strong></td>
<td>Account Management</td>
<td>Account Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accounts view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corporate Account editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Corporate Account activity</td>
</tr>
<tr>
<td></td>
<td>Business Partner Data</td>
<td>Business Partner Organization editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business Partners view</td>
</tr>
<tr>
<td><strong>Supplier Base</strong></td>
<td></td>
<td>New Supplier activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supplier editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Suppliers view</td>
</tr>
<tr>
<td><strong>Company Financial and Business Information by Nielsen Code</strong></td>
<td>Account Management</td>
<td>Corporate Account editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Corporate Account activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Account Overview fact sheet</td>
</tr>
<tr>
<td><strong>Point-of-Sale Transaction Lookup</strong></td>
<td>Account Management</td>
<td>Account Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td>Customer Invoicing</td>
<td>Point-of-Sale Transaction editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Point-of-Sale Transaction Overview fact sheet</td>
</tr>
<tr>
<td><strong>Product Web Search</strong></td>
<td>Sales Orders</td>
<td>Sales Order editor</td>
</tr>
<tr>
<td><strong>Reverse Lookup</strong></td>
<td>Account Management</td>
<td>New Partner activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contact editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corporate Account editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Corporate Account activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Private Account editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Private Account activity</td>
</tr>
<tr>
<td></td>
<td>Business Partner Data</td>
<td>New Business Partner Organization activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Business Partner Person activity</td>
</tr>
<tr>
<td></td>
<td>Liquidity Management</td>
<td>New Bank activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Clearing House activity</td>
</tr>
<tr>
<td></td>
<td>Marketing</td>
<td>New Competitor activity</td>
</tr>
<tr>
<td></td>
<td>Personnel Administration</td>
<td>New Social Insurance Fund activity</td>
</tr>
<tr>
<td></td>
<td>Supplier Base</td>
<td>Supplier editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Supplier activity</td>
</tr>
<tr>
<td></td>
<td>Tax Management</td>
<td>New Tax Authority activity</td>
</tr>
<tr>
<td>Port Binding</td>
<td>Work Center</td>
<td>Screen</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Reverse Lookup by Phone</td>
<td>Account Management</td>
<td>New Partner activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contact editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Contact activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corporate Account editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Corporate Account activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Private Account editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Private Account activity</td>
</tr>
<tr>
<td>Business Partner Data</td>
<td></td>
<td>New Business Partner Organization activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Business Partner Person activity</td>
</tr>
<tr>
<td>Liquidity Management</td>
<td></td>
<td>New Bank activity</td>
</tr>
<tr>
<td>Personnel Administration</td>
<td>Social Insurance Fund editor</td>
<td></td>
</tr>
<tr>
<td>Supplier Base</td>
<td></td>
<td>Supplier editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Supplier activity</td>
</tr>
<tr>
<td>Tax Management</td>
<td></td>
<td>New Tax Authority activity</td>
</tr>
<tr>
<td>Route Planner</td>
<td>Field Service and Repair</td>
<td>Order Pipeline view</td>
</tr>
<tr>
<td>Shipping Service</td>
<td>Outbound Logistics</td>
<td>Outbound Deliveries view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Outbound Delivery editor</td>
</tr>
<tr>
<td>Port Binding</td>
<td>Work Center</td>
<td>Screen</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>Social Network</td>
<td>Account Management</td>
<td>Account Overview fact sheet, Contact editor, Contact Overview fact sheet, Contacts view, New Contact activity, New Private Account activity, Private Account editor</td>
</tr>
<tr>
<td></td>
<td>Supplier Base</td>
<td>New Service Agent activity, New Supplier activity, Service Agent editor, Service Agent Overview fact sheet, Service Agents view, Supplier editor, Supplier Overview fact sheet, Suppliers view</td>
</tr>
<tr>
<td></td>
<td>Managing My Area</td>
<td>Employee Profile editor, Employee Profile Overview fact sheet</td>
</tr>
<tr>
<td>Sourcing</td>
<td>Purchase Requests and Orders</td>
<td>Purchase Request Assign Source of Supply, Purchase Requests view</td>
</tr>
<tr>
<td></td>
<td>Supplier Base</td>
<td>Supplier editor</td>
</tr>
<tr>
<td>URL Navigation</td>
<td>Account Management</td>
<td>Accounts view</td>
</tr>
<tr>
<td>Port Binding</td>
<td>Work Center</td>
<td>Screen</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><strong>Weather and Web Map</strong></td>
<td>Account Management</td>
<td>Partner Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Partner activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Partner editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Partners view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contacts view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contact Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contact editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Contact activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Account Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accounts view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corporate Account editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Corporate Account activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Private Account editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Private Account activity</td>
</tr>
<tr>
<td><strong>Business Partner Data</strong></td>
<td>Business Partner Contact Overview fact sheet</td>
<td>Business Partner Contact editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business Partner Organization editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business Partner Person editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business Partner Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business Partners view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee Contact Data fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Business Partner Organization activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Business Partner Person activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Business Partner Contact activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Registered Product activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Registered Product editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Registered Product Overview fact sheet</td>
</tr>
<tr>
<td><strong>Inbound Logistics</strong></td>
<td>Inbound Delivery Notification editor</td>
<td>New Inbound Delivery Notification activity</td>
</tr>
<tr>
<td><strong>Liquidity Management</strong></td>
<td>My Bank Overview fact sheet</td>
<td>My Banks view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>My Bank editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Bank activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clearing Houses view</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clearing House editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Clearing House activity</td>
</tr>
<tr>
<td><strong>Marketing</strong></td>
<td>Competitors view</td>
<td>New Competitor activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Competitor Overview fact sheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Competitor editor</td>
</tr>
<tr>
<td><strong>Outbound Logistics</strong></td>
<td>New Outbound Delivery activity</td>
<td>Outbound Delivery editor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Outbound Deliveries view</td>
</tr>
</tbody>
</table>

© 2012 SAP AG. All rights reserved.
### 6.7.2.3 Mashups and Web Services

#### Overview

Mashups are used to integrate data from SAP Business ByDesign with data provided by an online Web service or application. Users can access the content provided by these Web services and applications, and use it in their daily work. Mashups can include Web searches, company or industry business information, or online map searches.

#### Prerequisites

Configuration settings are normally performed by a key user. If you do not have the required authorization, contact your key user.

You have enabled Communities, Document Management and External Services in your system configuration. To find this business option, go to the Business Configuration work center and choose the Implementation Projects view. Select your implementation project and click Edit Project Scope. In the Scoping step of the project, ensure that People Collaboration, Intranet and External Services is selected within Communication and Information Exchange.

In the Questions step, expand the Communication and Information Exchange scoping element and select People Collaboration, Intranet and External Services. Select Communities, Document Management and External Services and answer the question related to Mashups, Web Services, and External Search Providers Integration.

Before a mashup can be used, it must be activated by a key user. This key user must have authorization for the Mashup Authoring and Mashup Web Services work center views.

Note that some mashups may not be applicable to your country. Your key user should ensure that the mashup is valid for your country before activating it for company-wide usage.

#### How to Get Mashups

Mashups can be provided in the following ways:

- A few preconfigured mashups are provided in SAP Business ByDesign.
- Your organization can procure mashups from the SAP Store. These mashup partner solutions are provided by SAP partners.
- The key user can create mashups for your organization.

**Preconfigured Mashups Provided in SAP Business ByDesign**

The following preconfigured mashups for Web services are provided in SAP Business ByDesign:

<table>
<thead>
<tr>
<th>Port Binding</th>
<th>Work Center</th>
<th>Screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Widget</td>
<td>Web widgets can be added to any screens including the Home work center Overview view as well as all other Overviews.</td>
<td></td>
</tr>
</tbody>
</table>
Preconfigured Mashups that Can Be Configured by the Key User

<table>
<thead>
<tr>
<th>Mashup Name</th>
<th>Mashup Type</th>
<th>Mashup Category</th>
<th>Port Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP (sap.com)</td>
<td>URL Mashup</td>
<td>News &amp; Reference</td>
<td>Search Provider</td>
</tr>
<tr>
<td>SAP Business ByDesign Community (sap.com)</td>
<td>URL Mashup</td>
<td>News &amp; Reference</td>
<td>Search Provider</td>
</tr>
<tr>
<td>Embedded Map (bing.com)</td>
<td>Custom Mashup</td>
<td>Location &amp; Travel</td>
<td>Web Map</td>
</tr>
<tr>
<td>Embedded Route (bing.com)</td>
<td>Custom Mashup</td>
<td>Location &amp; Travel</td>
<td>Route Planner</td>
</tr>
</tbody>
</table>

The key user can configure these mashups in the *Mashup Authoring* work center view in the *Application and User Management* work center.

In addition, the following preconfigured mashups are available for facilitating an automatic processing of customer payments using credit cards, in particular the card tokenization, authorization, and payment settlement with your acquirer. These mashups cannot be configured. For more information on these mashups, see Customer Payments by Credit Card.

Preconfigured Mashups that Cannot Be Configured by the Key User

<table>
<thead>
<tr>
<th>Mashup Name</th>
<th>Mashup Type</th>
<th>Mashup Category</th>
<th>Port Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payment Authorization Service (computop.com)</td>
<td>URL Mashup</td>
<td>Business &amp; Finance</td>
<td>Credit Card Authorization</td>
</tr>
<tr>
<td>Payment Authorization Service (paymetric.com)</td>
<td>URL Mashup</td>
<td>Business &amp; Finance</td>
<td>Credit Card Authorization</td>
</tr>
</tbody>
</table>

Mashup Partner Solutions Provided in the SAP Store

In addition, it is possible for your organization to procure mashup partner solutions from the SAP Store at [store.sap.com](http://store.sap.com). These solutions are then installed in your system. For more information, see SAP Store Quick Guide. Preconfigured mashups and mashup partner solutions are configured for use on selected screens in SAP Business ByDesign. For more information about the screens that are enabled for each mashup category, see Mashup Categories.

Creating Mashups for your Organization

The key user can create new URL, HTML, or data mashups based on a mashup category provided by SAP. For more information, see Mashup Authoring Quick Guide.

How to Use Mashups

After a mashup has been activated, it can be made visible on the screens for which it is configured. There are two ways this can be done:

- The key user does this centrally for all users in the *Adapt* menu. For more information, see Make a Mashup Visible on a Screen.
- The end user does this for himself or herself in the *Personalize* menu. For more information, see Personalize a Mashup.

1. If a mashup has not yet been activated, the key user can edit and activate it at a later date from the *Mashup Authoring* view.

How to Access Mashups

Once a mashup has been activated and made visible on screens, you can access it in the following ways:
A mashup can be displayed as an embedded screen. You can click a link on the screen. You can click Web Services and choose the mashup. You can use Enterprise Search and access external search providers that are integrated there.

Features

There are four different mashup types:

- **URL Mashups**
  A mashup that sends data from SAP’s on-demand solution to the URL of an online service provider. The service provider uses the data, for example, to perform a search, and the results are displayed in a new browser window.

- **HTML Mashups**
  A mashup that embeds an HTML or JavaScript based Web page directly on a screen.

- **Data Mashups**
  A mashup combines and displays data from both internal and external sources. To create a data mashup, an integrated authoring tool is used to transform or merge external Web services with internal business data using industry-standard Web service protocols.

- **Custom Mashups**
  A mashup that has been created as an add-on solution by SAP.

Key users can create new data, URL, and HTML mashups. Custom mashups can be created only by SAP.

In particular, mashups are distinguished by the following characteristics:

- **Mashup Categories**
  Mashups are grouped together into categories according to the type of service or information that they provide.

- **Port Binding**
  Certain fields on screens are predefined as screen out-ports and in-ports. These screen out-ports and in-ports are grouped together as a port binding.
  The port binding defines the type of information that can be used in the mashup and defines on which screens you can use the mashup. The mashup category specifies which port bindings are available.

For more information, see Mashup Categories.

API Keys and Authentication

- **API Keys**
  Some Web services require a unique API key to allow you to access their services. You can generate a company-specific API key on the service provider’s Web site and then use this key to authenticate access to the service from SAP’s on-demand solution.
  You can identify which mashups require an API key by the icon displayed in the API Keys column. The Enter API Keys dialog box displays the API keys that are required by the selected mashup.
  For a complete list of all SAP partners and their respective Web sites, see Web Services from our Partners in the Business Center.

- **Basic Authentication**
  Basic authentication allows you to enter a user name and password for authenticating access to a Web service. This user name and password is then used to automatically authenticate all other users accessing the service.

- **Secure Authentication**
For Web services that support the OAuth authentication protocol, you can set up secure authentication. The OAuth protocol enables secure, user-based access to Web services from SAP’s on-demand solution. Unlike other authentication methods, OAuth gives the user full control of his or her own authentication details. To use a mashup based on an OAuth authenticated Web service, users need to authenticate themselves on the service provider’s web site, where they are given a unique PIN code. This PIN code is then used to authenticate access to the service from SAP’s on-demand solution.

Key User Configuration

Key users can manage mashups in the following ways:

- Centrally from the Application and User Management work center
- Using the Adapt function on any screen enabled for mashups to access the adaptation sidecar

Key users can do the following tasks:

- Activate mashups for use by all users or deactivate unwanted mashups
  For more information, see Activate a Preconfigured Mashup.
- Activate mashup partner solutions from the SAP Store
  For more information, see Activate a Mashup Partner Solution.
- Change the way in which mashups are displayed, and test and publish the changes to all users
  For more information, see Make a Mashup Visible on a Screen.
- Create new mashups by copying an existing mashup and adapting the settings in the copied mashup
  For more information, see the Mashup Authoring Quick Guide.
- Add new Web services for use in data mashups
  For more information, see the Mashup Web Services Quick Guide.
- Create new URL, HTML, or data mashups based on a mashup category provided by SAP
  For more information, see the Mashup Authoring Quick Guide.

6.7.3 Tasks

6.7.3.1 Create a Port Binding for a Mashup

Overview

The SAP solution provides port bindings that specify on which screens mashups can be used. However, you can also create your own port bindings for your solutions, if required.

Prerequisites

You must have created a solution. The following prerequisites for creating solutions that use mashups apply:

- Mashups can be created for global solutions only. (Global solutions have names that do not start with “Z”, and are transported and deployed to customers). To create a global solution, ensure that the Create Sandbox Solution flag is not selected.

   It does not matter which type of solution you are using. It does not necessarily need to be of the type “mashup”.
You must enable the key user mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, in the Solution Explorer, right-click the solution and select Enable Key User. For more information about the key user mode, see Solution Explorer [page 56].

Before you can create a port binding, you need to create a port type package for the mashup inport. If the mashup is going to return data that is entered automatically into the SAP solution, then you need to create a port type package for the mashup outport as well. For information on creating port type packages, see Create a Port Type Package [page 526].

**Procedure**

1. Open your solution.
2. In the Solution Explorer, under Mashup Configuration, right-click Port Binding and select Create Port Binding.
3. Select the Mashup Port Binding template.
4. Enter a name for the port binding and click Add.
5. Right-click the port binding, and choose Open. The UI Designer opens, displaying a dialog box that contains the name of the port binding in the title.
6. In the Display Name field, enter the name of the port binding. In the Description field, enter a text that describes what the port binding is used for. This information is displayed in the dropdown list when you create a mashup and select a port binding.
7. In the Category field, select the mashup category to which the port binding should be assigned.
8. Select an inport type package from the dropdown list. Only your own inport type packages are displayed in the dropdown list.
9. Select an inport type reference. Only inport type references that have been defined for the port type package that you have selected are displayed.
10. If the mashup is going to return data to the SAP solution, select also an outport type package and an outport type reference.
11. Save and activate.

**Result**

You can now use the port binding when you create a mashup. For more information on creating mashups, see Creating Mashups [page 369].

**See Also**

Mashups Quick Guide [page 368]
6.7.3.2 Create a URL Mashup

Overview

In the software development kit (SDK), you can create URL mashups to send data from the SAP solution to a URL of a Web service provider. The results are displayed on the Web service provider’s Web site, which is opened in a new browser window.

Some Web services may pass your business data to a third-party organization, for example, account data is passed to a search engine when performing a reverse lookup in an online address book. We recommend that you check whether the mashup conforms to your company’s data privacy policies before activating the mashup.

Prerequisites

You know the end-point URL of the Web site providing the service. This could be, for example, the search results page of an online search provider.

- Mashups can be created for global solutions only. (Global solutions have names that do not start with “Z”, and are transported and deployed to customers). To create a global solution, ensure that the Create Sandbox Solution flag is not selected.
- It does not matter which type of solution you are using. It does not necessarily need to be of the type “mashup”.
- You must enable the key user mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, in the Solution Explorer, right-click the solution and select Enable Key User. For more information about the key user mode, see Solution Explorer [page 56].

Procedure

Open your solution. In the Solution Explorer, do one of the following steps:

- If you are creating a mashup for the first time for this solution, right-click Add-on Solution (In Development), then select Add New Item Create URL Mashup.
- If you have already created a mashup for this solution, you can simply right-click MashupAuthoring and select Create URL Mashup.

The SAP solution opens. If required, enter your user and password, then do the following steps:

1. Select a mashup category.
   The mashup category is used to group mashups by the type of information they provide. It is shown to business users when they personalize mashups in the side panel.
2. For the mashup category, you then select the port binding that you want to use in the mashup. This information is not shown to business users.
   The port binding defines which screen out-ports or in-ports can be used in the mashup and on which screens the mashup can be used.
3. Enter a name and a description for the mashup.
The mashup name will be displayed to business users when using the mashup.

*By default, the mashup has the status **Active**. This means it will be available for use when saved. If you want to prevent users from accessing the mashup immediately, change the status to **Inactive**.*

4. Under **URL Information**, enter the URL of the Web service and click **Extract Parameters**. The URL can be taken directly, for example, from the search results Web page of an online search provider. Copy and paste this URL directly into the field. The system then extracts all parameters from the URL and displays them in the table.

*Note that for URLs that do not use queries you can manually add curly brackets around terms that should act as placeholders. For example, in the URL [https://mail.google.com/mail/](https://mail.google.com/mail/#search/SAP), you can replace the word **SAP** with a search term in curly brackets, for example, **{term}**.

If you then enter [https://mail.google.com/mail/#search/{term}](https://mail.google.com/mail/#search/{term}) in the URL field and click **Extract Parameters**, the word in brackets is extracted as a parameter.

5. For any parameters that require a fixed value, check and adjust the value in the **Constant** column.

6. For dynamic parameters, in the **Parameter Binding** column, map the dynamic parameters to screen out-ports by selecting an out-port from the value help. The **Parameter Binding** dialog box displays a list of screen out-ports that have been configured for the selected mashup category and screen. Select the parameter you want to bind to the URL parameter. Note that if you are creating a mashup for the category route planner, you can also define the following optional settings:
   - **Code Format Conversion**: For parameters that contain a code, you can change how code formats are converted.
   - **Start Index for List**: You can define whether the array parameter starts with 0 or 1.
   - **Prefix for First Value/PREFIX for Next Value**: You can only define prefixes for URLs that have the index pattern $1 and for which the start index has been set to zero.

7. Click the **Preview** link to display the end result of the mashup.

8. Save your changes.

**Result**

When you return to the SDK, click **Refresh** to display the mashup in the Solution Explorer.

You can now do the following tasks:

- Add a Mashup to an SAP Floorplan  [page 397]
- Add a Mashup to Your Floorplan  [page 405]

The document **Add a URL Mashup** [page 120] provides an example of how you can add a URL mashup to a quick activity floorplan.

**See Also**

Mashups Quick Guide  [page 368]
6.7.3.3 Create a Data Mashup

Overview

In the software development kit (SDK), you can create data mashups for integrating data provided by third-party Web services with business data from the SAP solution.

Note that some Web services may pass your business data to a third-party organization, for example, account data is passed to a search engine when performing a reverse lookup in an online address book. It is recommended that you check whether the mashup conforms to your company’s data privacy policies before activating the mashup.

Prerequisites

You have either integrated a new Web service or have decided to base your mashup on one or more of the preconfigured Web services provided by SAP or partners.

Web services that you have integrated for use in data mashups must use the XML service protocol.

For more information about adding a new RSS/Atom, REST, or SOAP Web service, see Create a Mashup Web Service [page 393].

- Mashups can be created for global solutions only. (Global solutions have names that do not start with “Z”, and are transported and deployed to customers). To create a global solution, ensure that the Create Sandbox Solution flag is not selected.

- It does not matter which type of solution you are using. It does not necessarily need to be of the type “mashup”.

- You must enable the key user mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, in the Solution Explorer, right-click the solution and select Enable Key User. For more information about the key user mode, see Solution Explorer [page 56].

Procedure

Open your solution. In the Solution Explorer, do one of the following steps:

- If you are creating a mashup for the first time for this solution, right-click Add-on Solution (In Development), then select Add New Item Create Data Mashup.

- If you have already created a mashup for this solution, you can simply right-click MashupAuthoring and select Create Data Mashup.

The SAP solution opens. If required, enter your user and password, then do the following steps:

1. Select a mashup category.

   The mashup category is used to group mashups by the type of information they provide. It is shown to business users when they personalize mashups in the side panel.
2. For the mashup category, you then select the port binding that you want to use in the mashup. This information is not shown to business users.
   The port binding defines which screen out-ports or in-ports can be used in the mashup and on which screens the mashup can be used.

3. Enter a name and a description for the mashup.
   The mashup name will be displayed to business users when using the mashup.

   By default, the mashup has the status **Active**. This means it will be available for use when saved. If you want to prevent users from accessing the mashup immediately, change the status to **Inactive**.

4. On the **Define Mashup Details** step, create the mashup by adding building blocks to the authoring canvas and adjusting the properties. First of all, define the source of input for the mashup. You have the following options:
   **Mashup In-Ports**
   If mashup in-ports exist for the selected mashup category, the **Mashup In-Port** building block is added automatically to the authoring canvas. Click on the **Mashup In-Port** building block to view and adjust its parameters.
   **User Inputs**
   User inputs allow you to define a field in which a user can enter or select a value. This value is then used as input for the mashup. To add a user input to the mashup, select it from the pane on the left of the screen and drag and drop it onto the authoring canvas.

   **User Inputs**
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text Input</td>
<td>Allows users to enter a string of characters as input for the mashup.</td>
</tr>
<tr>
<td>Number Input</td>
<td>Allows users to enter a number as input for the mashup.</td>
</tr>
<tr>
<td>Select Option Input</td>
<td>Allows users to select an option from a dropdown list as input for the mashup.</td>
</tr>
</tbody>
</table>

   After adding a building block to the authoring canvas, click on the building block to view and adjust its properties. You can enter a name in the **Input Name** field. This can then be mapped to an input parameter, for example, in a search service. In the **Input Label** field, enter the field label that should be shown to users for this input field. For text and number inputs you can also define a default value, which can then be overwritten by other users. If you have selected a **Selection Option Input** building block, click the button next to **Options** to define the list of options that the user can choose from.

   **Data Sources**
   Data sources allow you to use data stored in a separate file as input for the mashup. This could be, for example, an XML file provided by an online service provider.

   **Data Sources**
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV Data Source</td>
<td>Use a CSV file as a input for the mashup. You can choose to either upload a file or reference an online CSV file.</td>
</tr>
<tr>
<td>XML Data Source</td>
<td>Use an XML file as input for the mashup</td>
</tr>
</tbody>
</table>

5. Select the services you want to integrate into the mashup.
   All Web services that have been configured and activated in the **Mashup Web Services** view are listed in the pane on the left of the screen and can be integrated into the mashup. These Web services are either SAP partner Web services or additional services you have created.
   Drag and drop an existing service onto the authoring canvas, or use the **New** menu to create a new Web service.

6. Using the mouse cursor, draw a connecting line between any mashup in-ports, user inputs, or data sources and the Web service.
This establishes the logical order of the mashup. It is possible to connect one mashup in-port with multiple services. After connecting the service to a user input or in-port, you can then edit the service parameters, if applicable.

7. Add actions to the mashup by dragging and dropping them onto the authoring canvas and drawing a connecting line between them and the service. Actions allow you to transform the content retrieved by the Web services. You have the following options:

### Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter Data</td>
<td>Filter the items returned by the service. You can create rules based on the parameters provided by the service.</td>
</tr>
<tr>
<td>Sort Data</td>
<td>Sort the display order of items in the results based on the parameters provided by the service.</td>
</tr>
<tr>
<td>Truncate Data</td>
<td>Truncate the number of items displayed in a feed after a given number.</td>
</tr>
<tr>
<td>Merge Data</td>
<td>Merge multiple feeds to display in a common results list. You can use this action to combine data from two sources that have the same format into a single results list.</td>
</tr>
<tr>
<td>Join Data</td>
<td>Join two data sources based on conditional rules to form a new data source. You can use this action to combine data from two sources that do not have the same format.</td>
</tr>
<tr>
<td>Add Field</td>
<td>Add one or more fields available in the parameters returned by the service to the results.</td>
</tr>
<tr>
<td>Rename Field</td>
<td>Rename one or more fields in the results.</td>
</tr>
<tr>
<td>Delete Field</td>
<td>Delete one or more fields from the results.</td>
</tr>
<tr>
<td>Extract Field</td>
<td>Extract a specific piece of information from the results, for example to use as input for another service within the same mashup.</td>
</tr>
<tr>
<td>Replace Text</td>
<td>Replace a specific piece of text in the results with a different text.</td>
</tr>
<tr>
<td>Convert Text</td>
<td>Convert text based on conditional rules. For example, by applying formatting to the text.</td>
</tr>
<tr>
<td>Concatenate Text</td>
<td>Concatenate text in the results. For example, you can use this action to concatenate two parameters in the search results to display in the same field.</td>
</tr>
<tr>
<td>Replace Value</td>
<td>Replace a value in the results based on conditional rules.</td>
</tr>
<tr>
<td>Arithmetic Operation</td>
<td>Perform an arithmetic operation with the results based on operation rules.</td>
</tr>
</tbody>
</table>

8. For mashup categories that allow the results data returned by the service to be integrated back into the screen, you can add a **Mashup Out-Port** building block to the mashup using drag and drop.

9. Click **Preview Result**.

To preview the mashup results you may have to manually add a value to the input field of the mashup in-port or user input. To do this select the building block and under properties, enter a value. In the runtime version, these values will be replaced automatically by either the parameter taken from the screen out-port or by the user entering their own value.

If you are using a service that uses the authentication method OAuth, then you will need to log on to the service and generate a PIN code in order to preview the service results. Afterwards when users are accessing the mashup, they will be prompted to create their own PIN code.

10. Adjust the display settings of the mashup results.

By default, the results are displayed in a table format. Click **Edit Display Settings** and select one or more formats in which the mashup will be made available to all users. You must specify a default display option. For each display option you can also configure additional display properties, such as the maximum number of items that should be displayed. Note that the XML view is provided for your information while creating the mashup but will not be shown to other users.

11. If required, you can also change the default refresh settings that are used to determine when the Web service used in the mashup is called. You have the following options:
12. On the Review step, you can view how the mashup will be displayed to users. Review your settings and click Finish to create the mashup.

**Result**

When you return to the SDK, click Refresh to display the mashup in the Solution Explorer.

You can now do the following tasks:

- Add a Mashup to an SAP Floorplan  [page 397]
- Add a Mashup to Your Floorplan  [page 405]

**See Also**

Mashups Quick Guide  [page 368]

### 6.7.3.4 Create an HTML Mashup

**Overview**

In the software development kit (SDK), you can create HTML mashups to embed an HTML or JavaScript based Web page into a screen.

Some Web services may pass your business data to a third-party organization, for example, account data is passed to a search engine when performing a reverse lookup in an online address book. We recommend that you check whether the mashup conforms to your company’s data privacy policies before activating the mashup.

**Prerequisites**

You know the URL of the Web site that you want to embed.

- Mashups can be created for global solutions only. (Global solutions have names that do not start with “Z”, and are transported and deployed to customers). To create a global solution, ensure that the Create Sandbox Solution flag is not selected.
  - It does not matter which type of solution you are using. It does not necessarily need to be of the type “mashup”.

- You must enable the key user mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, in the Solution Explorer, right-click the solution and select Enable Key User. For more information about the key user mode, see Solution Explorer [page 56].
Procedure

Open your solution. In the Solution Explorer, do one of the following steps:

- If you are creating a mashup for the first time for this solution, right-click Add-on Solution (In Development), then select Add New Item > Create HTML Mashup.
- If you have already created a mashup for this solution, you can simply right-click MashupAuthoring and select Create HTML Mashup.

The SAP solution opens. If required, enter your user and password, then do the following steps:

1. Select a mashup category.
   The mashup category is used to group mashups by the type of information they provide. It is shown to business users when they personalize mashups in the side panel.
2. For the mashup category, you then select the port binding that you want to use in the mashup. This information is not shown to business users.
   The port binding defines which screen out-ports or in-ports can be used in the mashup and on which screens the mashup can be used.
3. Enter a name and a description for the mashup.
   The mashup name will be displayed to business users when using the mashup.
   By default, the mashup has the status Active. This means it will be available for use when saved. If you want to prevent users from accessing the mashup immediately, change the status to Inactive.
4. Under Configuration Information, adjust the display height, as required. The default height is 327 pixels.
5. Specify the type of mashup you want to create. That is, select whether you want to enter HTML code or a URL.
6. Depending on what you have selected, do one of the following:
   - **Enter a URL**
     1. Enter a URL in the URL field and click [Extract Parameters].
        The system extracts the parameters from the URL and adds them to the table below, where you can edit constants and bind parameters to the screen out-ports available for the selected port binding.
        For URLs that do not use queries you can manually add curly brackets around terms that should act as placeholders. For example, in the URL `https://mail.google.com/mail/#search/SAP`, you can replace the word SAP with a search term in curly brackets, for example, `{term}`. If you then enter `https://mail.google.com/mail/#search/{term}` in the URL field and click [Extract Parameters], the word in brackets is extracted as a parameter.
        2. Click [Preview] to display the end result of the mashup.
   - **Enter HTML Code**
     1. In the HTML Code Editor you can copy or enter your own HTML code.
        In addition, you can use a third-party REST service or create your own REST service. We recommend that you use JSON service protocol for your REST service.
        To add the Web service to the HTML mashup, click [Add REST Service] and select a Web service from the REST Services dialog box. The code template of the API is inserted automatically in the HTML code editor.
2. Under **Input Parameters** and **Output Parameters**, the parameters are listed depending on the port binding that you have chosen. If required, you can copy the string code for a parameter to the HTML code editor by selecting the parameter and clicking [Copy]. The string code of the parameter is copied to the first row of the editor. You can then move this code string to the required location in the code.

3. Click [Preview] to display the end result of the mashup.

4. To test the mashup, you can enter sample values for the parameters, and click [Update Parameter Values] to the right of the HTML code editor.

   If you change the code, you need to click [Preview] again to display the updated result of the mashup.

7. Save your changes.

**Result**

When you return to the SDK, click [Refresh] to display the mashup in the Solution Explorer.

You can now do the following tasks:

- Add a Mashup to an SAP Floorplan  [page 397]
- Add a Mashup to Your Floorplan  [page 405]

**See Also**

Mashups Quick Guide  [page 368]

6.7.3.5 Example APIs for HTML Mashups

**Overview**

This document provides some example APIs to help you to code HTML mashups.

Any software coding or code lines/strings ("Code") provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

**External REST Service Call**

**Method**
sap.byd.ui.mashup.callService(serviceOptions)

**Usage**

When you click **Add REST Service** in the toolbar of the editor, the code template of this API is inserted automatically.
**Parameters**

serviceOptions is a JSON object with the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Mandatory/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>The ID of the REST service called</td>
<td>Mandatory</td>
</tr>
<tr>
<td>parameter</td>
<td>A JSON object that parameterizes the service</td>
<td>Optional</td>
</tr>
<tr>
<td>onComplete</td>
<td>Callback method name if the service call succeeds</td>
<td>Optional</td>
</tr>
<tr>
<td>onError</td>
<td>Callback method name if the service call fails</td>
<td>Optional</td>
</tr>
</tbody>
</table>

**Example**

```javascript
sap.byd.ui.mashup.callService({
  serviceId: 'CW00001',
  parameter: {'query': 'SAP'},
  onComplete: 'serviceCallback_CW00001',
  onError: 'exceptionHandler_CW00001'
});
```

**Responding to Application Context Update**

**Method**

`sap.byd.ui.mashup.onContextUpdate()`

**Usage**

If a port binding is assigned in the mashup, all the context parameters from out-port of the application and the pre-defined system parameters are displayed in the Input Parameters table. If you select an input parameter and click **Copy**, the system automatically copies the parameter in the selected row as a code string to the first line of the editor. You can then move this code string to the required location in the code.

After you have finished entering the code and clicked **Preview** to preview the results of the mashup, you can test the onContextUpdate API. Enter sample values for the input parameters and then click **Update Parameter Values**.

**Parameters**

The context can be accessed via `sap.byd.ui.mashup.context`, which is a JSON object with the following objects:

- **In-port**
  - if the mashup defines port binding, the values from the corresponding application out-port can be accessed by using `inport.ParameterName`.

- **System**
  - **Mashup system parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>system.LogonLanguage</td>
<td>Current display language in SAP Business ByDesign</td>
</tr>
<tr>
<td>system.ProductVersion</td>
<td>Current version of SAP Business ByDesign</td>
</tr>
<tr>
<td>system.ShortHostname</td>
<td>Short host name of SAP Business ByDesign system</td>
</tr>
</tbody>
</table>

**Example**

```javascript
sap.byd.ui.mashup.onContextUpdate = function() {
  var ctx = sap.byd.ui.mashup.context;
  // context parameters from ByDesign screen
  console.log('Logon language:' + ctx.system.LogonLanguage);
  // whole list of parameters for Port Binding Reverse Lookup
};
```
console.log('Company name:' + ctx.inport.CompanyName);
console.log('Address:' + ctx.inport.Address);
console.log('Street:' + ctx.inport.AddressStreet);
console.log('Street with house No.: ' + ctx.inport.AddressStreetHouseNumber);
console.log('House No.: ' + ctx.inport.AddressHouseNumber);
console.log('City:' + ctx.inport.AddressCity);
console.log('Country:' + ctx.inport.AddressCountry);
console.log('Country code:' + ctx.inport.AddressCountryCode);
console.log('Postal code:' + ctx.inport.AddressPostalCode);
console.log('State/province:' + ctx.inport.AddressStateProvince);
console.log('State/province code:' + ctx.inport.AddressStateProvinceCode);

Write Back Data to Application UI

Method
sap.byd.ui.mashup.fireResultUpdate(resultObject)

Usage
To consume this API, a writeback port binding (for example, Reverse Lookup) should be specified in the mashup. The Output Parameters table contains the parameters used to write back data to the corresponding application in-port. Click Copy to copy the corresponding parameter of the selected row to the editor. Using this API method, the preview values are filled with the values from the HTML application.

Parameters
resultObject is a JSON object which reflects the corresponding in-port of the application UI.

Example
// Example: Port Binding Reverse Lookup
// Address is a local object.
sap.byd.ui.mashup.fireResultUpdate({
  'CompanyName': address.name,
  'AddressStreetHouseNumber': address.street,
  'AddressStreet': address.street,
  'AddressCountryCode': address.country,
  'AddressStateProvinceCode': address.state,
  'AddressCity': address.city,
  'AddressPostalCode': address.zipCode
});

See Also
Create an HTML Mashup

6.7.3.6 Create a Mashup Web Service

Overview
In the software development kit (SDK), you can create mashup Web services that can then be integrated into a data or HTML mashup. RSS/Atom, REST, SOAP, and OData Web services are supported.
Prerequisites

- For RSS or Atom feeds, you know the end-point URL of the Web service or the WSDL location.
- For REST Web services, you need to know the request URL as well as any relevant parameters.
- For SOAP Web services, you need to know the WSDL location (URL).

Typically, you can find this information on the Web site of the service provider.

Procedure

Open your solution. In the Solution Explorer do one of the following steps:

- If you are creating a mashup Web service for the first time for this solution, right-click Add-on Solution (In Development), then select Add New Item Create .... Mashup Web Service (for example, Create REST Mashup Web Service).
- If you have already created a mashup Web service for this solution, you can simply right-click MashupAuthoring and select Create .... Mashup Web Service (for example, Create REST Mashup Web Service).

The SAP solution opens. If required, enter your user and password, then do the following steps:

1. Enter a name and description for the Web service.
2. Select whether you want the Web service to have the status Active or Inactive. Only active Web services can be used.
3. Depending on the type of mashup you are creating, do the following:

RSS or Atom Feed

a. On the New RSS/Atom Service screen, under Service Information, select an authentication method, if required.

   You can only use the OAuth standard to authenticate users accessing the service if OAuth is supported by the provider. You can find all of the information required on the provider’s Web site.

b. Enter the URL of the RSS or Atom feed and click Extract Parameters.

   The system extracts any parameters and displays them under Input Parameters. You can adjust these as required. Note that only those parameters that have empty constants can later be mapped to a mashup in-port or user input with default or dynamic values when creating a data mashup.

For URLs that do not use queries you can manually add curly brackets around terms that should act as placeholders. For example, in the URL https://mail.google.com/mail/#search/SAP, you can replace the word SAP with a search term in curly brackets, for example, {term}. If you then enter https://mail.google.com/mail/#search/{term} in the URL field and click Extract Parameters the word in brackets is extracted as a parameter.

REST

a. On the New REST Service screen, under Service Information, select an authentication method, if required.
You can only use the OAuth standard to authenticate users accessing the service if OAuth is supported by the provider. You can find all of the information required on the provider’s Web site.

b. Select an HTTP method.

c. Depending on the method you selected, do the following:

**If you selected GET:**
1. Enter the URL of the Web service and click **Extract Parameters**.
2. The system extracts any parameters and displays them under **Input Parameters**. You can adjust these as required.

   For URLs that do not use queries you can manually add curly brackets around terms that should act as placeholders. For more information, see the related note in the RSS or Atom Feed section above.

**If you selected POST:**
1. Select a content type. If you select **Form**, proceed as described above for the GET method. If you select **XML Body**, proceed as follows.
2. Enter the URL of the Web service and the XML code.
3. Click **Generate Body Parameters** to extract the parameters from the code, if applicable. The system extracts any parameters and displays them under **Input Parameters**. You can adjust these as required. Note that only those parameters that have empty constants can later be mapped to a mashup in-port or user input with default or dynamic values when creating a data mashup.

   You can also use curly brackets to define parameters in POST XML bodies. For example, you can replace a value returned by the REST service with a placeholder, for example `{placeholder}`, which can then be mapped to a screen out port in a mashup.

d. Under **Output Parameters**, click **Simulate** to preview the information that is returned by the Web service.

**SOAP**

a. On the **New SOAP Service** screen, under **Service Information**, enter the WSDL URL and click **Import**. The system extracts the WSDL content.

b. Select a service operation.

c. If required, click **Copy Description** to copy the service operation description into the mashup’s **Service Description** under **General Information**.

d. Enter constants, if required.

   Only those parameters that have empty constants can later be mapped to a mashup in-port or user input with default or dynamic values when creating a data mashup.

   The following restrictions apply when creating SOAP-based Web services: the XSD extensions union, list, and restriction are not supported; it is not possible to use header messages, imports in WSDLs, and maximum or minimum occurrence indicators; the XSD complex type structure is always handled as a sequence; for WSDLs containing more than one service, multiple ports with the same name are not supported; and SOAP 2.0 is not supported.

e. Under **Output Parameters**, click **Simulate** to preview what information will be returned by the Web service.

4. Save your changes.
Result

If the status of the Web service is active, it will be made available to integrate into a data or HTML mashup.
If the status of the Web service is inactive, you can edit and activate it at a later date from the Mashup Web Services view in SAP's on-demand solution.

See Also

Mashups Quick Guide  [page 368]
Using a Mashup Web Service in a Script File  [page 396]

6.7.3.7  Using a Mashup Web Service in a Script File

Overview

If you have configured a REST or RSS/Atom mashup Web service, you can implement the call of this Web service in a script file of your solution.

Prerequisites

- You have defined an active REST or RSS/Atom mashup Web service in your solution, either with basic authentication or without authentication.
- You have created a solution with at least one business object. In the example below, an action has been added to the business object definition.

Process Flow

1. Create a REST or RSS/Atom mashup Web service in your solution.
   For more information, see Create a Mashup Web Service  [page 393].
   After you have created the Web service, choose Simulate in the Mashup Web Services view to test it.
2. Create a script file for your business object, for example, an action.
3. Define the business logic to call the mashup Web service. The following is an example of an action.

```javascript
import AP.Common.GDT;
var serviceID : LANGUAGEINDEPENDENT_Text;
var parameter : NameAndValue;
var parameters : collectionof NameAndValue;
serviceID = "PW00001";
parameter.Name = "keywords";
parameter.Value = "Rome";
parameters.Add(parameter);
var result = WebServiceUtilities.ExecuteWebService(serviceID,
parameters);
var content = result.ResponseContent;
var returnCode = result.ReturnCode;
```
The parameters of this Web service call are as follows:

- **Input Parameters**
  - Service ID (mandatory)
    The ID of the REST or RSS/Atom mashup Web service of type `String`. Do not include the solution namespace. For example, if the service ID is displayed as follows: `YUYYRY2MY_MAIN/PW00001`, use only `PW00001` in the call.
  - Parameters (optional)
    Parameter of type `collectionof NameAndValue`. This refers to the table of input parameters defined in the REST mashup Web service.

- **Result**
  The type is `WSCallResult` and contains the following elements:
  - ReturnCode
    The return code of the Web service call as `String`.
  - ResponseContent
    The content of the response as `String`.

4. Save your script file.

### 6.7.3.8 Add a Mashup to an SAP Floorplan

**Overview**

When you add a mashup to an SAP floorplan, you can specify how it should appear on the screen. The following options are available:

- As an embedded pane
  This pane displays the mashup content directly in a section on the screen.
- As a new tab
- As a menu button
  The mashup can be accessed from `Web Services` located at the top of a screen.
- As a link in a section of a screen
  The link opens the mashup in a separate browser window.

The options that are available depend on the mashup itself and the anchors that the floorplan has. The following table lists the anchors and the possible appearances on the screen.

<table>
<thead>
<tr>
<th>Add Mashup Through</th>
<th>Appearance on Screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floorplan anchor</td>
<td>Embedded pane on a section of a screen</td>
</tr>
<tr>
<td></td>
<td>Menu button</td>
</tr>
<tr>
<td>Outport anchor</td>
<td>Embedded pane on a section of a screen</td>
</tr>
<tr>
<td></td>
<td>Tab</td>
</tr>
<tr>
<td></td>
<td>Menu button</td>
</tr>
<tr>
<td></td>
<td>Link in a section of a screen</td>
</tr>
<tr>
<td>Section group anchor</td>
<td>Embedded pane on a section of a screen</td>
</tr>
<tr>
<td></td>
<td>Link in a section of a screen</td>
</tr>
<tr>
<td>Switch navigation anchor</td>
<td>Tab</td>
</tr>
<tr>
<td>Overview page anchor</td>
<td>Embedded pane on a section of a screen</td>
</tr>
</tbody>
</table>
You can also choose an extension field to allow the user to toggle the display of the mashup on the screen by selecting and deselecting this extension field.

**Prerequisites**

- You have created the mashup. For more information, see:
  - Create a URL Mashup [page 384]
  - Create an HTML Mashup [page 389]
  - Create a Data Mashup [page 386]

- Mashups can be created for global solutions only. (Global solutions have names that do not start with “Z”, and are transported and deployed to customers). To create a global solution, ensure that the Create Sandbox Solution flag is not selected.

  It does not matter which type of solution you are using. It does not necessarily need to be of the type “mashup”.

- You must enable the key user mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, in the Solution Explorer, right-click the solution and select Enable Key User. For more information about the key user mode, see Solution Explorer [page 56].

- The SAP floorplan must have an existing outport with a stable anchor and a referenced anchor. You cannot create your own outports for SAP floorplans.

**Procedure**

1. **Open the SAP floorplan.**
   a. In the software development kit (SDK), right-click one of your floorplans in your solution, and select Open in UI Designer. The UI Designer opens.
   b. In the UI Designer, go to Configuration Explorer, and navigate to the SAP floorplan in which you want to add the mashup.
   c. Right-click the SAP floorplan, and click Open. The SAP floorplan opens in the Designer tab.

2. **Add the mashup to the SAP floorplan.**
   You can do this in the following ways:

   **Add Mashup Through a Floorplan Anchor**
   a. In the Configuration Explorer, open an SAP floorplan that has a floorplan anchor.
   b. Right-click the floorplan in the Designer tab, and choose Select Floorplan.
   c. In the Extensibility Explorer, under Operations, click Widget Mashups Management. A dialog box opens, and a list of available mashups is displayed.
   d. Select the required mashup.
   e. Select the checkbox in the Visible column. This allows the mashup to be displayed on the screen.
   f. Select the required appearance in the Appearance field.
   g. If a mashup should be displayed using the entire width of the screen, select Full Screen Width.
h. If there are extension fields of the type boolean, you can select an extension field in the **Visibility Binding** field, to allow the user to toggle the visibility of the mashup on the screen

i. Click **Apply**.  
The system creates a change transaction.

j. Save and activate your changes.

**Add a Mashup Through an Outport Anchor**

a. In the **Configuration Explorer**, open an SAP floorplan that has an outport anchor.

b. In the **Controller** tab, choose an outport under the **Outports** node.

c. In the **Extensibility Explorer**, under **Operations**, click **Mashups Management**.  
A dialog box opens, and a list of available mashups is displayed.

d. In the **Visible** column, select the checkbox for each mashup that you want to add to the floorplan.

   i. If the **Visible** column for a mashup is disabled, it means that the mashup is already available on the floorplan. To remove the mashup, you need to delete the corresponding change transaction in **Extensibility Explorer**.

   e. Select a mashup to configure it.  
The configuration details are displayed below.

   f. Select the required appearance in the **Appearance** field.

   g. If the mashup should be displayed using the entire width of the screen, select **Full Screen Width**.

h. If there are extension fields of the type boolean, you can select an extension field in the **Visibility Binding** field, to allow the user to toggle the visibility of the mashup on the screen

i. Click **Apply**.  
The system creates a change transaction.

j. Save and activate your changes.

**Add a Mashup Using a Section Group Anchor**

a. In the **Configuration Explorer**, open an SAP floorplan that has a section group anchor.

b. Select a section group in the floorplan.

c. In the **Extensibility Explorer**, under **Operations**, click **Mashups Management**.  
A dialog box opens, and a list of available mashups is displayed.

d. In the **Visible** column, select the checkbox for each mashup that you want to add to the floorplan.

   i. If the **Visible** column for a mashup is disabled, it means that the mashup is already available on the floorplan. To remove the mashup, you need to delete the corresponding change transaction in **Extensibility Explorer**.

   e. Select a mashup to configure it.  
The configuration details are displayed below.

   f. Select the required appearance in the **Appearance** field.

   g. Click **Apply**.  
The system creates a change transaction.

h. Save and activate your changes.

**Add a Mashup Through a Switch Navigation Anchor**

a. In the **Configuration Explorer**, open an SAP floorplan that has a switch navigation anchor.

b. Right-click the floorplan in the **Designer** tab, and choose **Select > NavigationItem**.

c. In the **Extensibility Explorer**, under **Operations**, click **Mashups Management**.
A dialog box opens, and a list of available mashups is displayed.

d. Select the required mashup.
   The configuration details are displayed below.

e. Select the checkbox in the Visible column. This allows the mashup to be displayed on the screen.
   
   If the Visible column for a mashup is disabled, it means that the mashup is already available on the floorplan. To remove the mashup, you need to delete the corresponding change transaction in Extensibility Explorer.

f. Select the required appearance in the Appearance field.

g. If a mashup should be displayed using the entire width of the screen, select Full Screen Width.

h. If there are extension fields of the type boolean, you can select an extension field in the Visibility Binding field, to allow the user to toggle the visibility of the mashup on the screen.

i. Under Parameter Binding, select the required extension fields to be bound to the parameters.
   
   Parameters that are marked with an asterisk (*) are mandatory.

j. If the mashup has an output parameter, select an extension field for output binding.

k. Click Apply.
   The system creates a change transaction.

l. Save and activate your changes.

**Add a Mashup Through an Overview Page Anchor**

a. In the Configuration Explorer, open an SAP floorplan that has an overview page anchor.

b. In the Extensibility Explorer, under Operations, click Add Content to Overview.
   A dialog box opens.

c. In the Type field, select Mashup from the dropdown list.
   The dialog box changes.

d. In the Mashup field, select the mashup that you want to add to the floorplan from the dropdown list, and click Apply.
   
   If the selected mashup is already part of the view, the Apply button is disabled.

e. If the mashup should be displayed using the entire width of the screen, select Full Screen Width.

f. Save and activate your changes.

**Result**

The mashup is now visible on the floorplan.

To remove a mashup from an SAP floorplan, you need to delete the corresponding change transaction in the Extensibility Explorer of the UI Designer, under Operations.

**See Also**

Mashups Quick Guide [page 368]
6.7.3.9 Add a URL Mashup to an SAP Floorplan

Overview

In this example you are going to add a URL mashup to the quick activity (QA) floorplan. This mashup will display the car park location on a map, based on the location information entered.

Prerequisites

- You have completed the example: Building a Solution [page 88]
- The mashup has been created in the SAP solution. For more information, see Mashups Quick Guide [page 368].

Procedure

1. Add an element on the root level of the Car Park business object to record location information.
   ```
   element CarParkLocation : LANGUAGEINDEPENDENT_Text;
   ```
2. Save and activate the BO.
3. Double-click the QA to open it in UI designer.
4. Click the Display<>Edit button to switch to edit mode.
5. Click the Update Metadata button to update the BO metadata.
6. Drag the CarParkLocation from the BO Browser / Data Model onto the Designer tab.
7. Change the label for the field to Car Park Location.
8. On the Controller tab, perform the following actions:
   a. Add a new outport with the name URL_Mashup_OutPort and the following settings:
      - PortType Package = /SAP_BYD_TF/Mashups/globalmashupsupporttypes
      - PortType Reference = Address_Info
   b. Bind the Address parameter to the CarParkLocation data field.
   c. In the Property Explorer, select the QA floorplan from the drop-down list at the top.
   d. Click the Anchor field under Extensibility.
      The Anchor Modelling dialog box appears.
   e. Click the plus sign to create a stable anchor for the floorplan.
f. In the Property Explorer, select the mashup outport from the drop-down list at the top.
g. Click the Anchor field under Extensibility. The Anchor Modelling dialog box appears.
h. Click the plus sign to create a stable anchor for the floorplan.
i. In the Property Explorer, click the Referenced Anchor field under Extensibility. The Anchor Modelling dialog box appears.
j. Click the plus sign to create a referenced anchor for the outport.
k. Click the three dots next to the XRep Path field.
l. In the Configured Anchors dialog box, click the anchor you created for the QA floorplan.
9. Save and activate.
10. Click the **Designer** tab.
11. Click the **Display<>Edit** button to switch to display mode.
12. In the **Extensibility Explorer**, select the mashup outport and click the **Mashups Management** button. The **Mashups Management** dialog box appears, listing the available mashups.
13. Select the mashup you want to use and click **Apply**.
14. Save and activate.

**Result**

The mashup is available from the *Web Services* menu on the QA. When you click the mashup, the URL is opened with the value entered into the *Car Park Location* field as an input parameter.
6.7.3.10 Add a Mashup to Your Floorplan

Overview

When you add a mashup to a floorplan that you have designed, you can specify how it should appear on the screen. The following options are available:

- As an embedded pane, with or without a port
  This pane displays the mashup content directly in a section on the screen.

- As a new tab

- A menu button in a function bar
  The mashup can be accessed from [Web Services] in the function bar at the top of a floorplan.

- A menu button in a list toolbar
  The mashup can be accessed from [Web Services] in the toolbar at the top of an object worklist.

- A link in a section of a screen
  This link opens the mashup in a separate browser window.

The options that are available depend on the mashup type. The following table lists the mashups and the possible appearances on the screen.
Mashups and Appearance on Screen

<table>
<thead>
<tr>
<th>Mashup Type</th>
<th>Appearance on Screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data mashups that write data back into the SAP</td>
<td>Embedded pane on a section of a screen, with or without a port</td>
</tr>
<tr>
<td>solution</td>
<td>Menu button to a function bar</td>
</tr>
<tr>
<td></td>
<td>Menu button to a list toolbar</td>
</tr>
<tr>
<td></td>
<td>Link in a section of a screen</td>
</tr>
<tr>
<td>URL mashups</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Data mashups that do not write data back into the</td>
<td>Embedded pane on a section of a screen</td>
</tr>
<tr>
<td>solution</td>
<td></td>
</tr>
<tr>
<td>HTML mashups</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Prerequisites**

- You have created the mashup. For more information, see:
  - Create a URL Mashup [page 384]
  - Create an HTML Mashup [page 389]
  - Create a Data Mashup [page 386]

- Mashups can be created for global solutions only. (Global solutions have names that do not start with “Z”, and are transported and deployed to customers). To create a global solution, ensure that the Create Sandbox Solution flag is not selected.

  - It does not matter which type of solution you are using. It does not necessarily need to be of the type “mashup”.

- You must enable the key user mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, in the Solution Explorer, right-click the solution and select Enable Key User. For more information about the key user mode, see Solution Explorer [page 56].

**Procedure**

1. Open your floorplan.
   - a. In the software development kit (SDK), in the Solution Explorer, right-click your floorplan and select Open in UI Designer.
      The UI Designer opens.
   - b. In the UI Designer, change to edit mode by clicking the Display <-> Edit button.

2. Add the mashup to your floorplan.
   Choose one of the following options to add a mashup to your floorplan:

   **Add a Mashup as a Link / Button in a Function Bar or a List Toolbar / Section of a Screen**
   - a. In the Controller, right-click Mashup Components, and select Add Mashup Component.
      A dialog box opens.
   - b. In the dialog box, go to the main folder of your solution and the folder where the mashups are stored.
   - c. Select a URL mashup or a data mashup that writes data back into the SAP solution, and click OK.
   - d. Go to the Toolbox tab to add a navigation item for a function bar or a list toolbar, or as a link to a section group.

   - If the Toolbox tab is not visible, make it visible by going to View Toolbox.
e. In the Controls section, drag and drop the required element (Link / Button / SectionGroup) to the floorplan.

f. Add an event argument with argument name $MashupPipeID, and a constant the target mashup component ID, for example, TM00020.

g. Create a new event handler, and add two operations: SyncDataContainer and FireOutport.

h. For FireOutport, select an available outport such as Address_Info_Out. If there is no outport for this floorplan, add a new outport in the Controller.

i. On the Properties tab, under Events, assign the event handler to a navigation item, for example, OnClick.

j. In the Simple Navigation tab, right-click on the Navigations folder, and select Add Navigation from the context menu.

k. Select the outport and the inport from the dropdown lists.

l. Bind the outport and inport parameters.

   The outport that you select here should be the same outport that you selected for FireOutport a few steps previously.

m. Save and activate your changes.

Add a Mashup With Port as an Embedded Pane

a. In the Configuration Explorer, select a mashup that is an embedded pane, and drag it to a section of the floorplan.

b. In the Controller, right-click Mashup Components, and select Add Mashup Component.

   A dialog box opens.

c. In the dialog box, go to the main folder of your solution and the folder where the mashups are stored.

d. Select a mashup that has the same pipe ID as the one you have just selected, and click OK.

e. In the Simple Navigation tab, right-click on the Navigations folder, and select Add Navigation from the context menu.

f. Select the outport and the inport from the dropdown lists.

g. If your floorplan does not have an outport, add an outport as follows:

   1. On the Controller tab, right-click Outports, and choose Add Outport. The new outport is displayed under Outports.

   2. In the Port Type Package field, click ... to select a port type package.

   3. In the dialog box, go to SAP_BYD_TF_Mashups, choose globalmashupporttypes, and click OK.

   4. In the Port Type Reference field, select a port type reference from the dropdown list.

   5. If there are parameters displayed in the table below, you can bind them. To do this, click ... next to Parameter Binding.

      If you do not bind the parameters, you will not be able to pass the values from the floorplan to the mashup.

   6. In the dialog box, choose the required element of the business object and click OK.

h. Bind the outport and inport parameters.

i. Save and activate your changes.

Add a Mashup Without Port as an Embedded Pane
a. In the **Configuration Explorer**, select a mashup that is an embedded pane, and drag it to a section of the floorplan.

b. Save and activate your changes.

---

### Add a Mashup as a Tab

a. Right-click the floorplan, and choose **Select NavigationItem**.

b. Under **Misc**, click **...** next to **NavigationItems** to configure the properties.

c. In the dialog box, select a view and click **Add**.

d. On the new view, add a mashup with or without a port as explained in the previous sections.

e. Save and activate your changes.

---

### Result

You can now test the mashup in your floorplan by clicking the **Run the Component** button (green triangle) in the toolbar.

---

### See Also

- **Example: Add a URL Mashup to Your Floorplan** [page 408]
- **Mashups Quick Guide** [page 368]

---

6.7.3.11 Example: Add a URL Mashup to Your Floorplan

---

### Overview

This example shows how to add a URL mashup as a link in a floorplan that you have designed.

---

### Prerequisites

- You have created the mashup. For more information, see:
  - **Create a URL Mashup** [page 384]
  - **Create an HTML Mashup** [page 389]
  - **Create a Data Mashup** [page 386]

- Mashups can be created for global solutions only. (Global solutions have names that do not start with “Z”, and are transported and deployed to customers). To create a global solution, ensure that the **Create Sandbox Solution** flag is not selected.

  It does not matter which type of solution you are using. It does not necessarily need to be of the type “mashup”.

- You must enable the key user mode for the solution in which you create the mashup to ensure that the system saves your content and any changes you make in the solution. To do this, in the **Solution Explorer**, right-click the solution and select **Enable Key User**. For more information about the key user mode, see **Solution Explorer** [page 56].
Procedure

1. **Add a link to your floorplan.**
   a. In the software development kit (SDK), in the Solution Explorer, right-click your floorplan and select Open in UI Designer. The UI Designer opens.
   b. In the UI Designer, change to edit mode by clicking the Display <-> Edit button.
   c. Go to the Toolbox tab.
   
   ![If the Toolbox tab is not visible, make it visible by going to View Toolbox](image)
   
   d. In the Controls section, drag and drop Link to the floorplan. The Link element is displayed on the floorplan. It contains a Link field and a Link hyperlink.
   e. You can rename the text of the Link field. To do this, ensure that the Link element is selected.
   f. On the Properties tab, under Text Information, click on the dropdown arrow in the Label field.
   g. In the dialog box, enter the required text, and click OK. The renaming of the hyperlink will be described later on in this document.

2. **Add your mashup.**
   a. In the UI Designer, go to the Controller tab of the floorplan.
   b. Right-click on Mashup Components, and select Add Mashup Component from the dropdown list.
   c. In the dialog box, go to the main folder of your solution and the folder where the mashups are stored. This name starts with MAIN_<Solution Name>_SRC_Mashups_Pipes_<Pipe ID>.
   d. Select your mashup as follows.
      • A URL mashup that is displayed in a new window
      • A data mashup that writes data back into the solution
   e. Click OK. The mashup is displayed on the Controller tab underneath Mashup Components.

3. **Add an event.**
   a. In the UI Designer, on the Designer tab, select the Link element in the floorplan.
   b. On the Properties tab, under Data Information, select EventArguments and click ...
   c. Click the Add Argument button. A new event argument is added.
   d. Under ArgumentName, rename the event argument to $MashupPipeID. We recommend that you copy the name $MashupPipeID from this document and paste it in the ArgumentName field.
   e. Leave the dialog box open, and go to the SDK.
   f. In the SDK, on the Solution Explorer tab, under MashupConfiguration>Mashup Authoring, select the mashup.
   g. On the Properties tab, copy the value in the Repository field. This value has the format MAIN_Solution Name>/<Pipe ID>. 
h. Go back to the open dialog box in the UI Designer, paste the value of the Repository field under Constant, and click OK.

i. Save your changes.

4. Create a new outport.
   a. On the Controller tab, right-click Outports and choose Add Outport.
   b. Right-click the outport and select Rename; then enter a new name for the outport.

5. Select a port type package for the outport.
   a. In the Port Type Package field, click ...
   b. In the dialog box, go to SAP BYD TF > Mashups, choose globalmashupporttypes, and click OK.

6. Configure a port type reference for the outport.
   a. In the Port Type Reference field, select a port type reference from the dropdown list.
   b. If there are parameters displayed in the table below, you can bind them. To do this, click [ ] next to Parameter Binding.
   c. In the dialog box, choose the required element of the business object and click OK.

7. Create an event handler.
   a. On the Controller tab, right-click Event Handlers and select Add EventHandler.
   b. Right-click the new event handler and select Rename; then enter a new name for the event handler, for example, MashupHandler.
   c. Under EventHandler Operations, click the Add Operation button twice to add two new operations.
   d. For each operation, under Type, click in the field and select the following operation types:
      ● SyncDataContainer
      ● FireOutport
   e. Under Name, you can change the names of these operation types
   f. For FireOutport, under Configure the operation, choose the outport that you created previously.
   g. Save and activate your changes.

8. Specify the navigation for the mashup.
   a. On the Controller tab, under Mashup Components, select the mashup.
   b. Go to the Simple Navigation tab.
   c. In the Navigations folder, right-click and select Add Navigation from the context menu. The Navigation Details tab opens.
   d. Select the outport and the inport from the dropdown lists.
      The source is the outport of your floorplan; the target is the inport of the mashup category.
   e. To map the outport and inport parameters from the floorplan to the mashup, select these, and click Bind.
      The results are displayed in the table below.
   f. Save and activate your changes.

9. Assign the event handler to the mashup in your floorplan
a. Go to the *Designer* tab.
b. Select the *Link* element in the floorplan.
c. On the *Properties* tab, under *Events*, select *OnClick*, and select the event handler that you created previously from the dropdown list.

10. **Add a data field to your business object.**
    a. Go to the *Data Model* tab.
    b. Right-click on your business object, and select *Add Data Field* from the context menu.
    The data field is added to your business object.
    c. Select *DataField*.
    d. On the *Properties* tab, under *Design*, enter a text for *Initial Value*.
    This text is used for the hyperlink that is displayed on the floorplan.

11. **Assign the data field to the mashup on your floorplan**
    a. Go to the *Designer* tab.
    b. Select the *Link* element.
    c. In the *Properties* tab, under *Data Information*, click the dropdown arrow next to *Text*.
    d. In the dialog box, go to the *Data Model* tab.
    e. Select *DataField* underneath your business object, and click *OK.*

12. **Save and activate your changes.**

**Result**

You can now test the mashup in your floorplan by clicking the *Run the Component* button (green triangle) in the toolbar.

**See Also**

- Add a Mashup to Your Floorplan  [page 405]
- Mashups Quick Guide  [page 368]

### 6.8 Print Forms

#### 6.8.1 Print Forms Quick Guide

You can create a print form to output the data from a business object that you created in your solution. You can define the data to include on the form, design the layout of the form, and select a floorplan to link to the form. Print forms are read-only portable document format (PDF) documents that are generated from data stored in the system. In the SAP Business ByDesign solution, print forms can be printed, sent as an e-mail attachment, or faxed to business partners.
Business and Technical Background

Business Objects

As part of your solution, you can create new business objects. For more information, see Business Objects Quick Guide [page 297].

Forms

Form templates are used to define the content and layout of documents that can be output from the system. For more information, see Forms.

SAP Business ByDesign Tools for Form Templates

In the SAP solution, customers use form templates to define the content and layout of documents that can be output from the system. For information about how customers work with the SAP tools for form templates, see the following quick guides:

- Form Template Maintenance Quick Guide
- Master Template Maintenance Quick Guide
- Form Template Selection Quick Guide

Tasks

Create a Print Form

For more information about creating a print form, see here [page 413].

Create a Preview Button

For more information about creating a preview button, see here [page 542].

Test a Print Form

If you have created BAC elements in your solution and assigned your solution content to a business option, you can perform scoping manually in the SAP Business ByDesign development environment. You must be assigned the Business User role and have access rights for the Business Configuration work center. For more information, see User Setup Quick Guide [page 39].

Alternatively, you can trigger the deployment of business configuration in the Solution Explorer by right-clicking the Business Configuration node and selecting Deploy Business Configuration.

1. When you use the Deploy Business Configuration function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the Deploy Business Configuration function deploys all business configuration content independently of any business options you have created.

To test a print form:

1. In the Solution Explorer, right-click the floorplan and click Preview Screen.

    If this is the first time you have selected this option, you may be prompted to enter a user name and password.
A preview of the screen is displayed.

2. On the screen, enter some values in the fields and click Save.

3. Click Preview.

If more than one template exists, a list of available templates is displayed in the Preview window. Select a template and click Open.

A preview of the selected print form is displayed as a PDF.

6.8.2 Associating an .xdp File with Adobe LiveCycle Designer

Procedure

1. In Windows, select Control Panel > All Control Panel Items > Default Programs.

2. Select Associate a file type or protocol with a program.

3. In the list, select the .xdp file type and select Change program....

4. Select Browse and then the FormDesigner.exe file in the folder where you installed the Adobe LiveCycle Designer program files. Click Open.

5. Make sure that the Adobe LiveCycle Designer ES entry and the Always use the selected program to open this kind of file checkbox are selected. Click OK.

When you now double-click an .xdp file to open it, the file opens in Adobe LiveCycle Designer.

6.8.3 Create a Print Form

Overview

You can create a print form to output the data from a business object. You can define the data to include on the form and design the layout of the form by using Adobe LiveCycle Designer.

Prerequisites

- Your solution contains an activated business object with at least one element.
- You have installed Adobe LiveCycle Designer.
- You have associated the .xdp file type with Adobe LiveCycle Designer.

For more information, see Associating an .xdp File with Adobe LiveCycle Designer [page 413].

Procedure

1. In the Solution Explorer, right-click the business object and select Create Print Forms.

   The Form Wizard opens.

2. In the Define Form Properties step, under Form Template Header, enter a Name and Description for the form template header.

3. Under Form Template Group, select a form template group (if available) or click New to create a new group.
4. If you selected to create a new group, in the **Form Template Editor** window, enter a name for the group in the **Name** field, click **Save**, and then click **Next**.
   
   The data structure of the business object is displayed.

5. In the **Define Form Data Structure** step, select the data elements to include on the form, and then click **Next**.

6. In the **Define Properties of Default Variant** step, enter a **Comment** to describe this form, select a **Language**, and then click **Next**.
   
   A summary of the settings you entered for the form is displayed.

7. In the **Review Form Settings** step, click **Finish**.
   
   The form is opened in Adobe LiveCycle Designer.

   If Adobe LiveCycle Designer does not open and an error message is displayed instead, make sure that you have associated the `.xdp` file type with Adobe LiveCycle Designer. For more information, see **Associating an .xdp File with Adobe LiveCycle Designer** [page 413].

8. Drag data elements from the **Data View** to the **Design View** to create your form.
   
   For more information about using Adobe LiveCycle Designer, see the Adobe product documentation.

9. Once you have designed the form, save your work and close Adobe LiveCycle Designer.

10. In the **Solution Explorer**, right-click the form group (`.ftgd`) file and select **Check In Content**. Once the check-in is complete, right-click the file and select **Activate**.

11. In the **Solution Explorer**, right-click the form header (`.fthd`) file and select **Check In Content**. Once the check-in is complete, right-click the file and select **Activate**.

**Result**

A print form is associated with the business object. You now need to make the form accessible to users from a floorplan. For more information about how to enable a preview function that allows users to preview the form as a PDF and provides options to save or print the form, see **Create a Preview Button** [page 542].

**See Also**

*Print Forms Quick Guide* [page 411]

6.9 Service Integration

6.9.1 Service Integration Quick Guide

In the SDK you can create synchronous or asynchronous message-based communication between a business object and another communication partner, such as a different system, by means of Web-service technology.

**Business and Technical Background**

**Web Services Quick Guide**

The SDK allows you to create a Web service on an SAP business object or on any of your business objects, which can then be consumed by a client application. This enables remote access to the business object data. You can
integrate an external Web service into your solution and address it in your business logic to enable access from the SAP solution to a remote application. You can also select a set of inbound and outbound services to create a communication scenario. This allows customers to create communication arrangements in the SAP solution to exchange business documents electronically. All communication for Web services is based on SOAP.

For more information, see here [page 480].

File Input Quick Guide

In the SAP solution the File Input Run view in the Application and User Management work center enables key users to create and schedule mass data runs for third-party data integration uploading files from other systems. The file input process allows business documents containing the files to be imported without the need for application-specific functionality. During a run, all files of the same type are input. The file input run receives inbound messages containing the files via application-to-application communication.

For more information about this tool in the SAP solution, see here.

Condition Evaluation

Condition evaluation is used to determine when a business object needs to send a message to the target business object, for example, a message to create or change a business object instance. For each business object identified as relevant for the subsequent service integration process, the system can evaluate a start condition, change condition and a cancel condition. By default, the system sends a message every time a business object is changed or deleted. To reduce the amount of messages, you can define that a message is raised only, for example, when the source business object has a specific status.

For more information, see here [page 416].

Tasks

Configure Internal Communication

You can use internal communication to exchange data between business objects that are located in different deployment units (DUs). To communicate, the business objects exchange XML messages in an asynchronous way which means that the source business object sends a request message to the target business object without expecting a response.

For more information about this task, see here [page 418].

Create XML File Input

In the SAP solution key users can use an XML file to import third-party data from other systems into a business object. To enable this functionality, you create an XML schema definition in the SDK. This schema definition allows customers to create an XML file, which can then be used to import the data into the SAP solution.

For more information about this task, see here [page 420].

Define Error Handling

Error and conflict handling is needed in message-based internal communication because there is no direct interaction with the user who initiated the process. Thus the receiver of a message has to handle the problem and has to trigger an appropriate resolution mechanism. A major part of development is to avoid error and conflict situations by design but it is not possible to avoid these situations entirely. This means that you also need to consider error and conflict situations in asynchronous communication.

For more information about this task, see here [page 421].
Implement the Condition Evaluation Logic

For internal communication, you can implement the business logic for condition evaluation to specify when a business object needs to send a message to the target business object, for example, a message to create or change a business object instance.

For more information about this task, see [Implement the Condition Evaluation Logic for Internal Communication](page 423).

6.9.2 Condition Evaluation for Internal Communication

Overview

Condition evaluation is used to determine when a business object needs to send a message to the target business object, for example, a message to create or change a business object instance. For each business object identified as relevant for the subsequent service integration process, the system can evaluate a start condition, change condition and a cancel condition. By default, the system sends a message every time a business object is changed or deleted. To reduce the amount of messages, you can define that a message is raised only, for example, when the source business object has a specific status.

Features

If you implement the evaluation logic for a condition, the system returns a Boolean value to express if the condition is true or false, for example, based on the status of the source business object.

| 1 | Condition evaluation checks the condition of the root node of a business object. |

Condition Types

You can define the following conditions:

- **Relevance Condition**
  Based on the data of the source business object, the system checks whether the business object instance is relevant for service integration. The system evaluates the relevance condition first. If the result is true, the message is sent to the business object and the system evaluates the start condition. If the result is false, the message is not sent and the service integration process is not started.

- **Start Condition**
  This condition starts the service integration process, if the business object instance was not yet involved in the service integration process or if the process was canceled. If the result is true, the system starts the service integration process, that is, it compiles and sends the message to the business object.

- **Change Condition**
  If the service integration process for the business object instance has already started, the system evaluates the change condition. If the result is true, that is, the business object instance has changed, the system sends a change message. If the result of the change condition is false, the system evaluates the cancel condition.

- **Cancel Condition**
  The system evaluates the cancel condition, if the result of the change condition is false.

Implementing the Evaluation Logic

When you create a condition, the system generates a script file. You implement the evaluation logic by using the scripting language. For more information, see [Implement the Condition Evaluation Logic for Internal Communication](page 423).
The following methods are available for service integration script files:

- **Return Statement**
  The return statement is mandatory for each condition because it returns the result of the evaluation. The argument is a variable or a literal of Boolean type, for example, `return true;`. For more information, see `Return Values` in *Syntax for Implementation of Actions and Events* [page 178].

- **GetProcessContext** Method
  The `GetProcessContext` method is called for business object nodes. The following change notification fields are available for this method:
  - `IsActiveInProcess`
    The business object node was already sent and not canceled.
  - `IsCreated`
    The business object node has been created.
  - `IsUpdated`
    The business object node has been updated.
  - `IsDeleted`
    The business object node has been deleted.

  For more information, see `Process Context` in *Syntax for Implementation of Actions and Events* [page 178].

- **InReconciliation** Parameter
  The `InReconciliation` parameter specifies that the service integration process is in reconciliation mode.
  For more information, see `Script File Parameter` in *Syntax for Implementation of Actions and Events* [page 178].

The following example shows the code for the implementation of a relevance condition, a start condition and a change condition:

- **Relevance Condition**
  
  ```java
  return true;
  ```

- **Start Condition**
  
  ```java
  return (this.SalesProcessingStatusCode == "2" ||
          this.SalesProcessingStatusCode == "3");
  ```

- **Change Condition**
  
  ```java
  return true;
  ```

The scenario for this implementation is as follows: Service integration is relevant if the sales process is started or finished. A start message will be sent if no message has been sent yet and the sales processing status is “In Process” (code value: 2) or “Finished” (code value: 3). A change message will be sent if the business object is changed and saved and the sales process is still in process or is finished. It is not necessary to implement a cancel condition because the change condition is always true.
6.9.3 Tasks

6.9.3.1 Configure Internal Communication

Overview
You can use internal communication to exchange data between business objects that are located in different deployment units (DUs). To communicate, the business objects exchange XML messages in an asynchronous way, which means that the source business object sends a request message to the target business object without expecting a response.

You have defined the Bonus Entitlement business object in the Customer Invoicing deployment unit. The Employee Compensation Agreement business object is located in the Human Capital Management (HCM) deployment unit. To transfer the bonus to the employees, you need to define an internal communication configuration between both business objects.

Prerequisites
You have created and activated your business object.

Procedure
1. In the Solution Explorer, right-click the business object in your solution and select Create Service Integration. The Service Integration Wizard appears.
2. In the Select Service Integration Type step, enter a name for your internal communication settings and select Internal Communication. Click Next.
3. In the General step, specify the settings as follows:

<table>
<thead>
<tr>
<th>Functions in the Wizard Step</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Integration Pattern</td>
<td>One-Way Notification</td>
</tr>
<tr>
<td>Triggered By</td>
<td>Offering Actor (Push)</td>
</tr>
<tr>
<td>Creation Type</td>
<td>Create New Service Integration</td>
</tr>
<tr>
<td>Execution Mode</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Transmission Mode</td>
<td>Complete</td>
</tr>
<tr>
<td>Prevent Data Deletion</td>
<td>Yes</td>
</tr>
<tr>
<td>Reconciliation</td>
<td>Yes</td>
</tr>
</tbody>
</table>

© 2012 SAP AG. All rights reserved.
Internal communication is executed in complete transmission mode, that is, the data is sent completely to the target business object. If data already exists in the target business object, this data is substituted by the data of the source business object. Data that is not part of the message is deleted in the target business object. You can prevent this by selecting the Prevent Data Deletion checkbox.

4. Click Next.

5. In the following steps, specify the actors of the communication. The offering actor is the source business object that sends the data. The consuming actor is the target business object that receives the data, that is, a business object in a different deployment unit. Both parties are represented by interaction roles, the offering interaction role and the consuming interaction role.
   a. In the Define Settings for Offering Actor step, specify the Offering Actor. In general, this is the business object that you selected when you started the wizard. Enter a name for the offering interaction role and click Next.
   b. In the Define Settings for Consuming Actor step, specify the Consuming Actor. You can select the business object, which represents the root node, or any other business object node. Enter a name for the consuming interaction role and click Next.

6. In the Message Interaction Structure and Identification step, specify the message definition and the identification attributes for the consuming actor as follows:
   a. Enter an interaction message name, for example, Maintain.
   b. Specify the structure of the message. Select the elements of the consuming actor node and its subnodes.
   c. In the Instance Identification screen area, choose an alternative key or a query. If you choose a query, for example, QueryByElements, choose the query parameters for the query mapping. Map each query parameter to a message field.
   d. Choose a subnode and a key field. A key field uniquely identifies a field of a business object instance. If needed, you can choose further subnodes and key fields. Click Next.

7. In the Interaction Message Mapping step, map the offering actor elements to the consuming actor elements. For example, map the EmployeeID element of the source business object Bonus Entitlement to the EmployeeID element of the Item node of the target business object Employee Compensation Agreement.
   To map an element, click a cell in the Offering Actor (Source) column. A dialog box appears in which you select the elements of the source business object.

8. Click Next.

9. In the Review step, check that the information you have entered is correct and click Finish.
   A new service integration definition with the file extension .pid appears in the Solution Explorer.

10. To activate your service integration settings, in the Solution Explorer, right-click the service integration definition and select Activate.

See Also

Define Error Handling  [page 421]
Implement the Condition Evaluation Logic  [page 423]
Service Integration Quick Guide  [page 414]
6.9.3.2 Create XML File Input

Overview

In the SAP solution key users can use an XML file to import third-party data from other systems into a business object. For more information, see File Input Quick Guide.

To enable this functionality, you create an XML schema definition in the SDK. This schema definition allows customers to create an XML file, which can then be used to import the data into the SAP solution.

Prerequisites

You have created and activated a business object.

Procedure

1. In the Solution Explorer, right-click your business object and select Create Service Integration. The Service Integration Wizard opens.
2. In the Select Service Integration Type step, enter a name for your XML schema definition and your file input settings and select XML File Input. Click Next.
3. In the Define Settings for Application Business Object step, specify the business object. In general, this is the business object that you selected when you started the wizard. However, you can also choose a different business object, for example, an SAP business object.
4. In the Code field, enter an alphanumeric code. The code can have up to five characters; letters need to be capitalized. Click Next.
5. In the File Format Definition and Identification step, proceed as follows:
   a. If you want to mass-enable the XML data structure so that it can be applied to multiple instances of the business object, select the List checkbox.
   b. To specify the data structure for the schema definition, select the elements of the business object and its subnodes.
   c. Choose a formatted ID. This ID is used for the identification of errors in error handling. If you selected the List checkbox previously, you do not have this option.
   d. In the Instance Identification area, choose an alternative key or a query. If you select a query, for example, QueryByElements, choose the query parameters for the query mapping. Map each query parameter to a message field.
   e. Choose a subnode and the corresponding key field. If needed, you can choose additional subnodes and key fields. Click Next.
6. In the Review step, check that the information you have entered is correct and click Finish.
   A new service integration definition with the file extension .pid appears in the Solution Explorer.
7. To activate your XML file input settings, in the Solution Explorer, right-click the service integration definition and select Activate.
   The system does the following:
It creates a folder within the standard WebDAV folder that is used by the SAP solution. The name of the new folder is as follows: \"your project name\\your XML file input name\". Customers need to use this folder at a later point in time for their XML file.

- It creates an XML schema definition.

To download the XML schema definition to your local PC, proceed as follows:

8. In the Solution Explorer, double-click your service integration definition. The service integration definition opens in a document window and the information you have defined is displayed on different tabs. On the Schema Definition tab, you can review the XML schema definition.

9. Select the General tab. Click the schema definition link and save the XML schema definition to your local PC. You can now predefine the business tasks that a key user can then monitor in the SAP solution if errors or conflicts occur during the upload of third-party data. For more information, see Define Error Handling [page 421].

If errors occur during the upload of third-party data to the SAP solution, for example, when invalid data violates the XML schema definition, the error messages are displayed in the system in the Application Log. When error messages are raised by the business object itself they are displayed in the Process Communication Errors view for key users, or tasks are created and assigned to the users responsible for the processing. The task are also displayed in the key user's Work view within the Application and User Management work center.

Result

Based on this XML schema definition, customers can create an XML file, for example, in Microsoft Office Excel. The XML file is to be stored in the WebDAV folder that was created by the SDK.

See Also

Service Integration Quick Guide [page 414]

6.9.3.3 Define Service Integration Error Handling

Overview

You can predefine the business tasks that a key user can monitor in the SAP solution if errors or conflicts occur during asynchronous communication. To do this, you define the tasks and assign them to the appropriate messages of the target business object.

In the SAP solution, the tasks will be assigned to the key user who is responsible for the support area that the target business object belongs to. Key users receive these tasks in the Process Communication Error subview of the Business Data Consistency work center view in the Application and User Management work center, if they choose Open Errors with Incidents in the Show dropdown list.

Prerequisites

In the SDK, you have created a service integration definition by using the Internal Communication wizard or the XML Input File wizard. For more information, see Configure Internal Communication [page 418] or Create XML File Input [page 420].
Procedure

1. In the SDK, open the solution that contains the service integration definition for which you want to configure the error handling.
2. In the Solution Explorer, double-click the *.pid file.
   The service integration definition opens in a document window.
3. Select the Error Handling tab.
   On this tab on the left, the message structure of the target business object is displayed. On the right, you can define tasks that correspond to business object messages. These tasks are the business tasks that will appear in the SAP solution.
4. In the Error Situations Solved by Task area, enter a short text for a task, for example, StartDateInvalid.
5. Assign the appropriate target business object messages to your task. To do this, select a message in the message structure on the left and then click Assign. You can assign one or more messages to each task. The messages then appear in the Assigned Messages area.
6. In the Error Situations Solved by Task area, click Long Text.
   A text editor opens.
7. Enter the details about the error situation and how to solve it and then click OK.
   The text editor closes and your text is displayed in the Long Text field.
8. Save your service integration definition.
9. In the Solution Explorer, right-click the *.pid file and select Activate.
   During activation, a task list view with the name of the service integration definition and the *.tasklistview file extension is created. You can only display the task list view in the user interface designer (UI designer).
10. In the Solution Explorer, right-click a screen file and select Open in UI Designer.
11. In the UI designer, in the Configuration Explorer, open the Process Communication Errors work center view.
    The view is located in the following folder: SAP_BYD_APPLICATION_UI/itsam/appmngt/pie/. The file name of the view is PCE_WCView.
12. In the Extensibility Explorer, add the task list view to the Process Communication Errors work center view as follows:
    1. Select the PCE_WCView.so.uianchor anchor and choose Assign Component To View.
    2. In the Add Assigned Object dialog box, select the TaskListView type.
    3. In the Component field, click ... and select your task list view. The view is located in the SRC subfolder of your project folder. For example, if the name of your project is A2BCD300, the folder name is MAIN_A2BCD300/SRC/.
    4. Click Apply.
13. Save and activate the PCE_WCView file.

See Also

Service Integration Quick Guide [page 414]
6.9.3.4 Implement the Condition Evaluation Logic for Internal Communication

For internal communication, you can implement the business logic for condition evaluation to specify when a business object needs to send a message to the target business object, for example, a message to create or change a business object instance.

For more information, see Condition Evaluation for Internal Communication [page 416].

Prerequisites

In the SDK, you have created a service integration definition by using the Internal Communication wizard. For more information, see Configure Internal Communication [page 418].

Procedure

1. In the SDK, open the solution that contains the service integration definition for which you want to implement the condition evaluation.
2. In the Solution Explorer, open the service integration definition by double-clicking the *.pid file. The service integration definition opens in a document window.
3. On the Conditions tab, select a condition, for which you want to implement the evaluation logic, for example, Relevance Condition, and click Add.
   The SDK generates a script file that opens in a separate document window.
4. In the code editor, define the evaluation logic for the condition. The condition must contain at least a return statement that returns the result of the condition evaluation with a Boolean return value. For more information, see Service Integration Methods [page 199].
5. Optional: Add further conditions and define the respective evaluation logic.
6. Save the service integration definition. Then right-click the service integration definition and select Activate Script Files.
   The script files for the implementation of the condition evaluation logic are activated.

   The script files do not appear in the Solution Explorer. However, you can only select Activate Script Files if script files exist.

See Also

Service Integration Quick Guide [page 414]
6.10 Tracing

6.10.1 Tracing Quick Guide

You can use tracing to record information about your code as it is being executed. From the Trace Explorer, you can run a local trace, run a trace for an end user, use incident traces for troubleshooting purposes, and view the results of a recorded trace.

You can access Tracing as follows:

- To enable the Trace Explorer tool window, click View ➔ Other Windows ➔ Trace Explorer. The Trace Explorer tool window provides a view of active and completed traces.
- To enable the Trace Explorer toolbar, click View ➔ Toolbars ➔ Trace Explorer. The Trace Explorer toolbar provides buttons that allow you to activate and deactivate a trace.

Business and Technical Background

Tracing

You can use tracing to record information about your code as it is being executed, including variable values, informational messages, and error messages. By inspecting the information recorded by a trace, you can obtain a better understanding about how your code works, troubleshoot problems, and debug your code. Tracing is available to developer users in the Trace Explorer.

For more information about tracing, see here [page 425].

Trace Class and Methods

You can use the methods defined by the Trace class in the ABSL namespace to record information about the state of your code. Method calls from the Trace class are evaluated and recorded when a trace is active in the system. The Trace class provides methods to record informational messages (Trace.Info) and error messages (Trace.Error).

For more information about the Trace class and methods, see here [page 427].

Tasks

Run a Local Trace

You can activate tracing on any system that you can log on to with your developer user account. When a trace is active, code snippets are logged as they are executed, and methods from the Trace class are evaluated and recorded.

For more information about this task, see here [page 429].
## Run a Trace for a Business User

You can activate tracing for a business user. When tracing has been activated for a business user, the code that is executed by that user is traced. Because tracing is a developer feature, business users cannot enable tracing directly.

For more information about this task, see [here](#) [page 429].

## Working with Incident Traces

When a business user (key user or end user) on a productive system reports an incident, you can ask them to reproduce the incident and include trace information. This information can help you track down the source of the issue that the business user is experiencing. Incident traces contain the same information as local traces, but are tied to the life cycle of the incident.

For more information about this process, see [here](#) [page 430].

## View Trace Results

When a trace has been completed, you can view the recorded information in the Trace Explorer.

For more information about this task, see [here](#) [page 431].

### 6.10.2 Business and Technical Background

#### 6.10.2.1 Tracing

## Overview

You can use tracing to record information about your code as it is being executed, including variable values, informational messages, and error messages. By inspecting the information recorded by a trace, you can obtain a better understanding about how your code works, troubleshoot problems, and debug your code. Tracing is available to developer users in the Trace Explorer.

When a trace is active, some information is traced automatically, including the identity of each code snippet that is executed. You can enhance the information that is recorded during a trace by adding methods from the Trace class to your code snippets. This way, you can record informational and error messages, and output variable values.

## Prerequisites

You are familiar with creating business objects and working with the scripting language.

## Features

### Local Traces

You use local traces to trace code on a system that you can access with your developer user account. You can use local traces to debug code that is in development, or to investigate issues that have been reproduced on a local system. Developer users can manually activate and deactivate local traces in the Trace Explorer.

- **Local Traces for Developer Users**
  
  When you activate a trace, the code snippets that you run under your developer user account are recorded and the methods from the Trace class are evaluated and recorded.
Local Traces for Business Users

You can configure tracing to record the actions of a business user instead of tracing the actions performed under your developer user account. This enables you to debug workflows that cannot be performed by a developer user in the Trace Explorer, but can be performed by a business user in the SAP solution. Business users cannot activate tracing directly because tracing is a developer feature that is accessed only through the Trace Explorer in the SDK, which business users cannot access. When you run a trace for a business user, the code snippets that are executed by the business user are recorded and the trace statements in the code are evaluated and recorded.

For more information, see Run a Trace for a Business User [page 429].

Incident Traces

When business users encounter errors in a productive system, they can use the incident reporting process to report the error and request assistance. Business users can attach trace information to incidents, and you can use this information to help troubleshoot the cause of the incident.

You work with incident traces the same way as you work with local traces. However, incident traces are tied to the life span of an incident. Incident traces are created when the incident is submitted and are deleted when the incident is closed. When you use the trace information that is attached to an incident, it is important to compare it to code from the same version of the system that was used to report the incident. Otherwise, the code paths may have changed and trace statements may be different.

For more information, see Working with Incident Traces [page 430].

Performance

Trace information is only recorded while an active trace is running. When tracing is not active, no system resources are allocated for tracing. The methods from the Trace class are only evaluated when a trace is active; therefore you can include them in your code snippets without incurring any performance penalty during the course of regular operation.

Interpreting Trace Results

In the Trace Explorer window, you can view information about the session, sequence, and code snippets that are called, as well as the results of evaluating methods from the Trace class that have been added to code snippets, and information about other errors that have occurred.

As each session, sequence, and code snippet is called, a trace message is recorded in a tree structure in the trace details pane. The trace messages are numbered in the order that they were called, and are nested under the calling parent session, series or code snippet as follows:

- **Session**
  Each session is displayed in order.

- **Sequence**
  Each sequence is displayed in order.

- **Code snippet**
  Each code snippet is displayed in the order it was called, and may be nested if it was called from another code snippet. This view displays a summary of all of the trace messages for this code snippet. If any of the trace messages for the code snippet are error messages, then an error icon is displayed. This enables you to quickly...
identify which code snippets contain error messages without having to manually view the details about each code snippet.

You can view details about the trace messages recorded for each code snippet by double-clicking the code snippet entry in the Trace Explorer window. This displays the code snippet in the Trace Details pane. The Trace Details pane contains a list of the trace messages that were recorded for the code snippet. The first item in this list is a message that identifies the code snippet. The rest of the list contains the results of calling methods from the Trace class, and errors that may have occurred. The details for the trace messages are as follows:

- **Status**
  This defines whether the message is an error or for information purposes.

- **Type**
  This defines the type of message. For example, every code snippet is identified by a message whose type is “this” object ID. Messages that were generated by method calls from the Trace class are identified by a message of type user-defined trace command.

- **Number**
  The order that this trace message was processed for this code snippet.

- **Name**
  The name of the message. For messages recorded by the Trace class, this displays the first parameter passed to the Trace.Info and Trace.Error method calls.

- **Value**
  The value of this message. For object identification messages, this can be the alternative key or the UUID of the object. For Trace class messages, this is the value of the second parameter passed to Trace.Info and Trace.Error method calls.

See Also

Trace Class and Methods [page 427]
Tracing Quick Guide [page 424]

6.10.2.2 Trace Class and Methods

Overview

You can use the methods defined by the Trace class in the ABSL namespace to record information about the state of your code. Method calls from the Trace class are evaluated and recorded when a trace is active in the system. The Trace class provides methods to record informational messages (Trace.Info) and error messages (Trace.Error).

Trace Class

The Trace class provides methods that you can use to record information about the state of your code.

Importing the Trace Class

You import the ABSL namespace to access the Trace class.
import ABSL;

Trace.Error Method

You use the Error method to record information about errors. There are two overloads of the Error method, taking one or two String parameters. The results of evaluating these methods are displayed in the Trace Details pane and are marked by an error icon.

| Syntax              | Trace.Error(String);  
|                     | Trace.Error(String, String); |
| Example             | Trace.Error(“Creation failed”);  
|                     | Trace.Error(“Creation failed”, this.Identifier.Content); |

Trace.Info Method

You use the Info method to record information that is not an error case. There are two overloads of the Info method, taking one or two String parameters. The results of evaluating these methods are displayed in the Trace Details pane and are marked by an information icon.

| Syntax              | Trace.Info(String);  
|                     | Trace.Info(String, String); |
| Example             | Trace.Info(“Test triggered”);  
|                     | Trace.Info(“Test triggered”, this.Identifier.Content); |

Example

import ABSL;  
Trace.Info(“Start Test”, this.Identifier.Content);  
Trace.Error(“Invalid parameter”, this.Identifier.Content);  
Trace.Info(“End Test”);

See Also

Tracing [page 425]  
Tracing Quick Guide [page 424]
6.10.3 Tasks

6.10.3.1 Run a Local Trace

Overview
You can activate tracing on any system that you can log on to with your developer user account. When a trace is active, code snippets are logged as they are executed, and methods from the Trace class are evaluated and recorded.

Prerequisites
Before you run a local trace, you have to configure the SDK to view the Trace Explorer toolbar and the Trace Explorer window.
You must know how to run your code snippets, and you may want to add methods from the Trace class to your code snippets to record custom information.

Procedure
1. To begin recording trace information, on the Trace Explorer toolbar click the Activate Trace button.
2. Run the code snippets that you want to trace.
3. To stop recording trace information, on the Trace Explorer toolbar click the Deactivate Trace button.
4. To view traces, in the Trace Explorer window click the Local Traces tab.
5. To update the list of traces, click the Refresh Traces button.
The completed trace appears in this list, identified by the time that trace was started, and the ID of the user executing the code.

See Also
Run a Trace for a Business User  [page 429]
Trace Class and Methods  [page 427]

6.10.3.2 Run a Trace for a Business User

Overview
You can activate tracing for a business user. When tracing has been activated for a business user, the code that is executed by that user is traced. Because tracing is a developer feature, business users cannot enable tracing directly.
You may want to trace a workflow that can be run by a business user but cannot be run by a developer user. For example, you must be logged on to the SAP solution as a business user to access some workflows. Since business users cannot activate and deactivate tracing, developer users can run a trace on their behalf through the Trace class.
Explorer. When a developer user activates a trace for a business user, the code that is executed by the business user is traced.

Prerequisites
Before you run a local trace, you have to enable the Trace Explorer toolbar and the Trace Explorer window. To record custom information, you have to add methods from the Trace class to your code snippets.

Procedure
1. To open the trace settings dialog, click Tools Options SAP-General Tracing.
2. Select Enable tracing for business user.
3. In the text box, enter the ID of the business user to enable tracing. For example, enter PGREENE to enable tracing for the business user Peter Greene.
4. On the Trace Explorer toolbar, click the Activate Trace button. A dialog is displayed that confirms tracing has been started for the business user.
5. Log on to the system as the business user and run the workflow you want to trace.
6. On the Trace Explorer toolbar, click the Deactivate Trace button.
7. In the Trace Explorer window, click the Local Traces tab.
8. To update the list of traces, click the Refresh Traces button. The trace appears with the time stamp of when the trace was started and the ID of the business user.

See Also
Run a Local Trace [page 429]
Trace Class and Methods [page 427]

6.10.3.3 Working with Incident Traces

Overview
When a business user (key user or end user) on a productive system reports an incident, you can ask them to reproduce the incident and include trace information. This information can help you track down the source of the issue that the business user is experiencing. Incident traces contain the same information as local traces, but are tied to the life cycle of the incident.

Prerequisites
You use the same version of the system that was used to report the incident.

Process Flow
1. An end user encounters an error in the SAP Business ByDesign system and reports an incident to a key user.
2. The key user returns the incident and requests trace information.
3. In the Incident editor, the end user clicks [Start Trace] and repeats the steps that caused the incident.
4. In the SDK, the incident trace appears in the Incident Trace tab of the Trace Explorer.
5. Developer users with the relevant authorization can view the incident trace in the Trace Explorer.
6. The incident trace remains accessible until the incident is closed.

See Also

Working with Incidents
Tracing [page 425]

6.10.3.4 View Trace Results

Overview

When a trace has been completed, you can view the recorded information in the Trace Explorer.

Prerequisites

A completed local trace or open incident trace exists.

Procedure

1. To view the Trace Explorer window, click View Other Windows Trace Explorer. The Trace Explorer window appears.
2. In the Trace Selection area, select the type of trace to view:
   - To view local traces, click the Local Traces tab.
   - To view traces for open incidents, click the Incident Traces tab
3. To display the current list of completed traces, click the Refresh button.
4. Select the check box next to the traces that you want to view, and click the Get Trace Details button. Each selected trace appears in its own tab.
5. Expand the trace details to show the sessions, sequences, and code snippets that have been recorded for the trace.
   If the code snippet contains any error messages, it is marked by an error icon. This enables you to easily identify the code snippets that contain errors. If the code snippet contains no errors, it is marked by an information icon.
6. Double-click a code snippet trace message to see detailed trace information for that code snippet in the Trace Details pane.

See Also

Tracing [page 425]
Run a Local Trace [page 429]
Run a Trace for a Business User [page 429]
6.11 User Interface Custom Panes and Custom Controls

6.11.1 Custom Panes and Custom Controls Quick Guide

You can create custom panes and custom controls to provide a unique view of the data in your business objects. You create a custom pane or a custom control by using Microsoft Visual Studio 2010 and the Software Development Kit (SDK) for Custom Panes and Custom Controls to create a Microsoft Silverlight component that can be embedded in screens and other user interface (UI) components.

You use the SDK to create an embedded component that provides a container for the custom pane, and provides bindings that map to the data fields of the custom pane.

You use the user interface designer (UI designer) to add the embedded component to screens and bind the fields of the custom pane to data fields in your business object.

Business and Technical Background

Custom Panes

Custom panes enable you to create UI components that provide a unique view of the data in your business object. You use custom panes to present a view of your data that cannot be created from the UI components available in the UI designer.

Custom panes must be uploaded to an embedded component before they can be added to screens with the UI designer. This embedded component provides bindings that map data fields in the custom pane to the data fields of your business object.

Custom Controls

A custom control is a way to provide a control inside a form pane, list pane-column, or identification region. It can be used in cases where the standard framework controls do not meet your requirements. The custom control adheres to the layout rules of form panes, such as label handling, list panes, and so on.

Application Programming Interface (API) for Software Development Kit (SDK) Custom Panes and Custom Controls

The API for SDK Custom Panes and Custom Controls contains class methods that allow you to implement your custom pane so that it can integrate with the user interface of the SAP solution.

For more information about this SDK, see here [page 433].

Tasks

Create a Custom Pane or Custom Control

You use Microsoft Visual Studio 2010 to create custom panes and custom controls as Microsoft Silverlight components. You use the classes and methods defined by the Application Programming Interface (API) for Software Development Kit (SDK) Custom Panes and Custom Controls to implement your component so that it can be included in the interface of the SAP solution.
For more information about this task, see [here](#) [page 472].

**Create an Embedded Component for a Custom Pane**

You use an embedded component to provide a container for a custom pane. The embedded component provides bindings that access the data fields of the custom pane, and can be bound to data fields in a business object. You can add an embedded component directly to screens.

For more information about this task, see [here](#) [page 473].

**Add an Embedded Component to a Screen**

You can add an embedded component to a screen, and then bind the fields of the embedded component to the data fields of a business object.

For more information about this task, see [here](#) [page 474].

**Test Your Custom Pane**

1. In the **Solution Explorer**, right-click your screen and select **Preview Screen**.
2. Verify the custom pane is displayed
3. Verify the behavior of your custom pane. For example, if you have created **BeforeSave** and **AfterModify** events, enter some data and save the screen. Check that the fields update as expected.

**See Also**

[Example: Best Employee Graph](#) [page 476]

6.11.2 Business and Technical Background

6.11.2.1 Application Programming Interface (API) for SDK Custom Panes and Custom Controls

**Overview**

Using the User Interface Designer, partner developers can model user interfaces (UIs) on SAP Business ByDesign. To ensure consistency across UIs, modeling is defined by SAP Business ByDesign standards. In cases where these standards do not fulfill your requirements, you can use custom panes.

A custom pane is derived from an abstract framework class. It is an empty pane that can be filled with a customer Microsoft Silverlight user interface and can access the controller and data container of the running modeled UI component.

Using the API, you can access `Controller.DataContainer` to read the data from the actual running client component and register to events in case of changes. You configure data retrieval as you do for normal modeled user interfaces. In your data model, you bind to business objects and through controller operations, you trigger business object operations, such as read, edit, and, create, queries, and actions.

A custom control is on the granularity of a form pane field, the control of a list column. It comes with the label inside the form and aligns as any other control in the form pane. Using a custom control, you can set custom visualization.

Using the API, you can visualize data on your UI in the manner you choose.
<table>
<thead>
<tr>
<th>Namespaces</th>
<th>Classes and Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SDK</strong> [page 435]</td>
<td>Class  SDKCusomControl  [page 435]</td>
</tr>
<tr>
<td></td>
<td>Class  SDKCusomPane  [page 437]</td>
</tr>
<tr>
<td></td>
<td>Interface  ISDKCusomControlAPI [page 439]</td>
</tr>
<tr>
<td></td>
<td>Interface  ISDKCusomPaneAPI [page 439]</td>
</tr>
<tr>
<td></td>
<td>Interface  ISDKDaAPI  [page 440]</td>
</tr>
<tr>
<td></td>
<td>Class  SDKCusomCustomValueChangedEventArgs  [page 440]</td>
</tr>
<tr>
<td></td>
<td>Class  SDKPaneConfigAttribute  [page 441]</td>
</tr>
<tr>
<td><strong>Common.Controller</strong> [page 441]</td>
<td>Interface  IApplicationMessage  [page 442]</td>
</tr>
<tr>
<td></td>
<td>Class  ValidationFailedEventArgs  [page 443]</td>
</tr>
<tr>
<td><strong>Common.Controller.Data</strong> [page 443]</td>
<td>Class  CodeListLoadingEventArgs  [page 444]</td>
</tr>
<tr>
<td></td>
<td>Interface  IBaseDataElement  [page 444]</td>
</tr>
<tr>
<td></td>
<td>Interface  ICodeListItem  [page 445]</td>
</tr>
<tr>
<td></td>
<td>Class  ValueFormattingResult  [page 445]</td>
</tr>
<tr>
<td><strong>Common.Controller.Event</strong> [page 446]</td>
<td>Class  EventHandlerFiredEventArgs  [page 447]</td>
</tr>
<tr>
<td></td>
<td>Interface  IEventHandler  [page 447]</td>
</tr>
<tr>
<td></td>
<td>Interface  IEventHandlers  [page 448]</td>
</tr>
<tr>
<td></td>
<td>Interface  IOperationState  [page 448]</td>
</tr>
<tr>
<td></td>
<td>Class  PortFiredEventArgs  [page 448]</td>
</tr>
<tr>
<td></td>
<td>Class  NavigationParameters  [page 450]</td>
</tr>
<tr>
<td></td>
<td>Class  NavigationTargetDetails  [page 451]</td>
</tr>
<tr>
<td></td>
<td>Class  ParamValue  [page 451]</td>
</tr>
<tr>
<td><strong>Common.Controls</strong> [page 452]</td>
<td>Interface  IAbstractBaseControl  [page 452]</td>
</tr>
<tr>
<td></td>
<td>Interface  ILabelControl  [page 453]</td>
</tr>
<tr>
<td></td>
<td>Interface  IParameter  [page 454]</td>
</tr>
<tr>
<td></td>
<td>Interface  IPlaceableField  [page 454]</td>
</tr>
<tr>
<td><strong>Controller</strong> [page 455]</td>
<td>Interface  ISDKComponentController  [page 455]</td>
</tr>
<tr>
<td><strong>Controller.Data</strong> [page 456]</td>
<td>Interface  ISDKBindingScope  [page 456]</td>
</tr>
<tr>
<td></td>
<td>Interface  ISDKBindingScopeItem  [page 457]</td>
</tr>
<tr>
<td></td>
<td>Interface  ISDKCodeList  [page 457]</td>
</tr>
<tr>
<td></td>
<td>Interface  ISDKDataContainer  [page 459]</td>
</tr>
<tr>
<td></td>
<td>Interface  ISDKDataField  [page 460]</td>
</tr>
<tr>
<td></td>
<td>Interface  ISDKDataRecord  [page 461]</td>
</tr>
<tr>
<td></td>
<td>Interface  ISDKDataRecordList  [page 462]</td>
</tr>
<tr>
<td></td>
<td>Interface  ISDKDataRecordListRow  [page 464]</td>
</tr>
</tbody>
</table>
6.11.2.2 SDK Namespace

6.11.2.2.1 SDK

Classes and Interfaces

<table>
<thead>
<tr>
<th>Namespace</th>
<th>Classes and Interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller.Property</td>
<td>Interface: ISDKBoundProperty [page 465]</td>
</tr>
<tr>
<td></td>
<td>Class: ISDKDataRecordEventArgs [page 467]</td>
</tr>
<tr>
<td></td>
<td>Class: ISDKDataRecordListEventArgs</td>
</tr>
<tr>
<td></td>
<td>Class: ISDKDataRecordListRowChangeEventArgs [page 468]</td>
</tr>
<tr>
<td></td>
<td>Class: ISDKListRoundtripPendingEventArgs [page 468]</td>
</tr>
<tr>
<td></td>
<td>Class: ISDKStatesChangedEventArgs [page 469]</td>
</tr>
<tr>
<td></td>
<td>Class: ISDKValueChangedEventArgs [page 469]</td>
</tr>
<tr>
<td>Controls</td>
<td>Interface: ISDKControlFactory [page 470]</td>
</tr>
<tr>
<td>Model</td>
<td>Interface: ISDKTextPool [page 471]</td>
</tr>
</tbody>
</table>

6.11.2.2.2 SDKCustomControl

Overview

An abstract base class from which your own implementation must derive.

A custom control is a way to provide a control inside a form pane, list pane-column, or identification region. It can be used in cases where the standard framework controls do not meet your requirements. The custom control adheres to the layout rules of form panes, such as label handling, and list panes, and so on. It comes with some intrinsic framework behavior.

The following chart displays the inheritance diagram for SDKCustomControl.
The inheriting object points to its source.

**Public Member Functions**

- **void** `Initialize(ISDKCustomControlAPI [page 439] in_value)`
  Used to initialize the custom pane implementation. This method should only be called by the framework.

- **string** `LookupParameter(string name)`
  Returns the value of a custom parameter (if available), returns null if no such parameter value is available.

- **virtual void** `SetValue(string rawValue)`
  Use this method to override if the bound value is to be set with the given raw value.

- **virtual void** `SetEnabled(bool isEnabled)`
  Use this to react on enabled state changes by the framework.

- **virtual void** `SetReadOnly(bool isReadOnly)`
  Use this to react on read only state changes by the framework.

- **virtual void** `SetVisible(bool isVisible)`
  Use this to react on visible state changes by the framework.

- **virtual void** `SetMandatory(bool isMandatory)`
  Use this to react on mandatory state changes by the framework.

**Static Public Attributes**

- static readonly DependencyProperty `ValueProperty`
  Metadata properties for control-binding in the ByDesign table.

- static readonly DependencyProperty `ReadOnlyProperty`
  Metadata properties for control-binding in the ByDesign table.

- static readonly DependencyProperty `EnabledProperty`
  Metadata properties for control-binding in the ByDesign table.

**Protected Member Functions**

- **virtual void** `OnInitialize` (page 439)
  Once the custom control starts, custom initialization can be put in here in the derived implementation.

- **abstract void** `OnTerminate` (onTerminate)
  On shutdown of the control in the onTerminate, custom cleanup can be performed.

- **void** `NotifyValueChange(string newValue)`

**Protected Attributes**

- **ISDKCustomControlAPI [page 439]** `mHostData`
  This grants access to all framework features to access, for example, the component controller or have a way to create own placeable fields through the control factory.
Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>ExtensionObject [get]</td>
</tr>
<tr>
<td>string</td>
<td>AutomationIdPrefix [get]</td>
</tr>
<tr>
<td>ISDKComponentController [page 455]</td>
<td>Controller [get]</td>
</tr>
<tr>
<td>List&lt;IParameter [page 454]&gt;</td>
<td>Parameters [get]</td>
</tr>
<tr>
<td>CustomControlType</td>
<td>FieldDefinition [get]</td>
</tr>
<tr>
<td>new FrameworkElement</td>
<td>FrameworkElement [get]</td>
</tr>
<tr>
<td>bool</td>
<td>IsVisible [get]</td>
</tr>
</tbody>
</table>

Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new EventHandler&lt; EventArgs&gt;</td>
<td>SizeChanged [get]</td>
</tr>
<tr>
<td>Event&lt; EventArgs&gt;</td>
<td>VisibilityChanged [get]</td>
</tr>
<tr>
<td>SDKCustomControlValueChangedEventHandler</td>
<td>ValueChanged [get]</td>
</tr>
</tbody>
</table>

See Also

SDK [page 435]

6.11.2.2.3 SDKCustomPane

Overview

An abstract base class from which your own implementation must derive.

A custom pane is a way to provide a canvas area inside a floorplan implemented with native Microsoft Silverlight controls. It can be used in cases where the standard framework controls do not meet your requirements.

The paradigm is that the visualization is controlled by the custom pane implementation but data retrieval happens through the modelled bindings of the component. Inside the custom pane implementation, the component controller and its data container can be accessed, thereby identifying the data to be displayed inside the custom pane.

The following chart displays the inheritance diagram for SDKCustomPane.
The inheriting object points to its source.

Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>LookupParameter</td>
<td>string LookupParameter (string name) Returns the value of a custom parameter (if available), returns null if no such parameter value is available.</td>
</tr>
<tr>
<td>void</td>
<td>Initialize</td>
<td>void Initialize (ISDKCustomPaneAPI in_value) Used to initialize the custom pane implementation. This method should only be called by the framework.</td>
</tr>
</tbody>
</table>

Protected Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>virtual void</td>
<td>OnInitialize</td>
<td>virtual void OnInitialize Once the custom pane starts, custom initialization can be put in here in the derived implementation.</td>
</tr>
<tr>
<td>abstract void</td>
<td>OnTerminate</td>
<td>abstract void OnTerminate On shutdown of the custom pane in the onTerminate, custom cleanup can be performed.</td>
</tr>
</tbody>
</table>

Protected Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISDKCustomPaneAPI</td>
<td>mHostData</td>
<td>ISDKCustomPaneAPI mHostData This grants access to all framework features to access, for example, the component controller, or have a way to create own placeable fields through the control factory.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>ExtensionObject</td>
<td>object ExtensionObject [get]</td>
</tr>
<tr>
<td>string</td>
<td>AutomationIdPrefix</td>
<td>string AutomationIdPrefix [get] This property should be used while setting unique automation IDs for controls in CUBB/Custom Pane.</td>
</tr>
<tr>
<td>ISDKControlFactory</td>
<td>ControlFactory</td>
<td>ISDKControlFactory ControlFactory [get] Reference to the control factory.</td>
</tr>
<tr>
<td>ISDKComponentController</td>
<td>Controller</td>
<td>ISDKComponentController Controller [get] Reference to the controller of the component.</td>
</tr>
<tr>
<td>List&lt; IParameter &gt;</td>
<td>Parameters</td>
<td>List&lt; IParameter &gt; Parameters [get] List of custom parameters from the designtime model.</td>
</tr>
<tr>
<td>ISDKBindingScope</td>
<td>BindingScope</td>
<td>ISDKBindingScope BindingScope [get] Binding scope set at creation time, which needs to be passed eventually to the control factory if placeable fields needs to be created in the context of a list or nested list.</td>
</tr>
<tr>
<td>bool</td>
<td>IsOnPage</td>
<td>bool IsOnPage [get] Determines whether the custom page is still on the page (events could be triggered asynchronously).</td>
</tr>
</tbody>
</table>
6.11.2.2.4 ISDKCustomControlAPI

Overview
An interface for custom controls.
The following chart displays the inheritance diagram for ISDKCustomControlAPI.

The inheriting object points to its source.

Properties

<table>
<thead>
<tr>
<th>CustomControlType</th>
<th>FieldDefinition [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Access to the modeled field definition of the custom control.</td>
</tr>
</tbody>
</table>

6.11.2.2.5 ISDKCustomPaneAPI

Overview
An interface for a custom pane to allow access to the control factory and the binding scope if available.
The following chart displays the inheritance diagram for ISDKCustomPaneAPI.

The inheriting object points to its source.

Properties

<table>
<thead>
<tr>
<th>ISDKControlFactory [page 470]</th>
<th>ControlFactory [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The control factory allows creating placeable fields which are intrinsic framework controls defined in the model but without any placement in the modelled UI.</td>
</tr>
</tbody>
</table>
ISDKBindingScope [page 456]  BindingScope [get]  
The binding scope needs to be passed in cases where the placeable fields to be created eventually should act in the context of a list or nested list.

See Also
SDK [page 435]

6.11.2.2.6  ISDKDataAPI

Overview
Data services for extension projects. It contains access to the component controller, configuration parameters, and automation ID.

The following chart displays the inheritance diagram for ISDKDataAPI.

The inheriting object points to its source.

Properties

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>AutomationIdPrefix [get]</td>
<td>Automation ID prefix for all child controls to support automation.</td>
</tr>
</tbody>
</table>

See Also
SDK [page 435]

6.11.2.2.7  SDKCustomControlValueChangedEventArgs

Overview
Event arguments for the value change of the custom control.
Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDKCustomControlValueChangedEventArgs (String value)</td>
<td></td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Value [get]</td>
<td>Contains the changed value.</td>
</tr>
</tbody>
</table>

See Also

SDK [page 435]

6.11.2.2.8 SDKPaneConfigAttribute

Overview

The following chart displays the inheritance diagram for SDKPaneConfigAttribute.

```
SAP.UI5.UI5.UI SDK SDKPaneConfigAttribute
  SAP.UI5.UI5.UI Common Extensibility.MEP.Export ExtensionMetadata
  SAP.UI5.UI5.UI Common Extensibility.MEP.Export ExtensionConfigAttribute
```

The inheriting object points to its source.

Public Member Functions

SDKPaneConfigAttribute (string fullTypeName, string inAssemblyName)

See Also

SDK [page 435]

6.11.2.3 Common.Controller Namespace

6.11.2.3.1 Common.Controller

Namespaces

- Data
Classes and Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>IApplicationMessage [page 442]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>ValidationFailedEventArgs [page 443]</td>
</tr>
</tbody>
</table>

Enumerations

**SAP.BYD.LS.UI.SDK.Common.Controller.MessageSeverity**

Enumerator:
- Success
- Info
- Warning
- Error
- Other

**SAP.BYD.LS.UI.SDK.Common.Controller.RoundtripPendingState**

Enumerator:
- None
- Reload
- Paging
- AddRow
- RemoveRow
- ExpandAll

6.11.2.3.2 IApplicationMessage

Overview

An interface for an application message.

Properties

<table>
<thead>
<tr>
<th>MessageSeverity</th>
<th>Severity [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Description [get]</td>
</tr>
<tr>
<td>string</td>
<td>Key [get]</td>
</tr>
<tr>
<td>string</td>
<td>MessageKey [get]</td>
</tr>
</tbody>
</table>

Returns the severity level enum of the message.

Description of the message.

Unique message key to differentiate messages. The backend returns a separate message for each affected field. For these rows, the key has to return the same value.

The key of the message.
The help link of the message.

See Also
Common.Controller [page 441]

6.11.2.3.3 ValidationFailedEventArgs

Overview
Event argument for validation failed event.

Public Member Functions
ValidationFailedEventArgs (string message)

Properties
string Message [get]
Returns the message of the validation failed event.

See Also
Common.Controller [page 441]

6.11.2.4 Common.Controller.Data Namespace

6.11.2.4.1 Common.Controller.Data

Classes and Interfaces

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeListLoadingEventArgs</td>
<td>[page 444]</td>
</tr>
<tr>
<td>IBaseDataElement</td>
<td>[page 444]</td>
</tr>
<tr>
<td>ICodeListItem</td>
<td>[page 445]</td>
</tr>
<tr>
<td>ValueFormattingResult</td>
<td>[page 445]</td>
</tr>
</tbody>
</table>

Enumerations


Enumerator:
- Standard

Enumerator:
- LoadPending
- LoadFinished


Enumerator:
- Complete
- Incomplete
- Complete_Invalid
- Incomplete_Invalid

6.11.2.4.2 CodeListLoadingEventArgs

Overview
Event arguments for a code list loading event.

Public Member Functions

CodeListLoadingEventArgs (CodeListLoadingState state)

Properties

<table>
<thead>
<tr>
<th>CodeListLoadingState</th>
<th>State [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the state of the code list loading event (LoadPending or LoadFinished).</td>
</tr>
</tbody>
</table>

See Also

Common.Controller.Data [page 443]

6.11.2.4.3 IBaseDataElement

Overview
A basic data element interface, IBaseDataElement is a base interface for all data container elements, such as data records, data record lists, and data fields.
The following chart displays the inheritance diagram for IBaseDataElement.
The inheriting object points to its source.

### See Also

- Common.Controller.Data [page 443]
- ISDKDataField [page 460]
- ISDKDataRecord [page 461]
- ISDKDataRecordList [page 462]
- ISDKDataRecordListRow [page 464]

#### 6.11.2.4.4 ICodeListItem

**Overview**

An interface for a single code list item.

**Properties**

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Key [get]</td>
<td>The key of the code list item.</td>
</tr>
<tr>
<td>string</td>
<td>Code [get]</td>
<td>The code of the code list item.</td>
</tr>
<tr>
<td>string</td>
<td>Value [get]</td>
<td>The value of the code list item.</td>
</tr>
<tr>
<td>string</td>
<td>VisualKey [get]</td>
<td>The formatted value of the code list item.</td>
</tr>
</tbody>
</table>

### See Also

- Common.Controller.Data [page 443]

#### 6.11.2.4.5 ValueFormattingResult

**Overview**

This class holds information about the success and failure of value formatting.
Public Member Functions

`ValueFormattingResult` (bool success, string message)

Static Public Attributes

<table>
<thead>
<tr>
<th>static <code>ValueFormattingResult</code></th>
<th><code>SUCCESS</code> = new <code>ValueFormattingResult</code> (true, null)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Standard success instance.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>bool</th>
<th><code>Success</code> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates if the value formatting is successful.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th><code>Message</code> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the message of the value formatting.</td>
</tr>
</tbody>
</table>

See Also

Common.Controller.Data [page 443]

6.11.2.5 Common.Controller.Event Namespace

6.11.2.5.1 Common.Controller.Event

Classes and Interfaces

<table>
<thead>
<tr>
<th>Class</th>
<th>EventHandlerFiredEventArgs [page 447]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>IEventHandler [page 447]</td>
</tr>
<tr>
<td>Interface</td>
<td>IEventHandler [page 448]</td>
</tr>
<tr>
<td>Interface</td>
<td>IOperationState [page 448]</td>
</tr>
<tr>
<td>Class</td>
<td>PortFiredEventArgs [page 448]</td>
</tr>
</tbody>
</table>

Enumerations

SAP.BYD.LS.UI.SDK.Common.Controller.Event.EventTriggerTypes

Enumerator:

- Generic
- CancelOnLocalValidationError

SAP.BYD.LS.UI.SDK.Common.Controller.Event.OperationStates

Enumerator:

- Unknown
- Enabled

© 2012 SAP AG. All rights reserved.
6.11.2.5.2 EventHandlerFiredEventArgs

**Overview**

Event arguments for fire-event events.

**Public Member Functions**

*EventHandlerFiredEventArgs* (string name)

Constructs a new event handler instance.

**Properties**

<table>
<thead>
<tr>
<th>string</th>
<th><strong>HandlerName</strong> [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Event handler name of the source event.</td>
</tr>
</tbody>
</table>

**See Also**

[Common.Controller.Event](#) [page 446]

6.11.2.5.3 IEventHandler

**Overview**

An interface for a single event handler defined in the model.

**Public Member Functions**

<table>
<thead>
<tr>
<th>IOperationState [page 448]</th>
<th>GetOperationStates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the list of operations defined for the event handler.</td>
</tr>
</tbody>
</table>

**Properties**

<table>
<thead>
<tr>
<th>UXEventHandlerType</th>
<th><strong>Definition</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The event handler’s definition from the model.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>isEnabled</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Determines whether the event handler is enabled. For example, the metadata of a BO action can set the <code>isEnabled</code> property.</td>
</tr>
</tbody>
</table>

**See Also**

[Common.Controller.Event](#) [page 446]
6.11.2.5.4  IEventHandlers

**Overview**

An interface for collection of event handlers of a component model.

**Public Member Functions**

<table>
<thead>
<tr>
<th>IEventHandler</th>
<th>GetList</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gets the list of event handlers.</td>
</tr>
</tbody>
</table>

**See Also**

Common.Controller.Event  [page 446]

6.11.2.5.5  IOperationState

**Overview**

An interface for an operation state.

**Properties**

<table>
<thead>
<tr>
<th>UXEventHandlerOperationType</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[get] References the source operation for the operation state.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OperationStates</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[get] Returns the current operation state of the operation (Unknown, Enabled, Disabled).</td>
</tr>
</tbody>
</table>

**See Also**

Common.Controller.Event  [page 446]

6.11.2.5.6  PortFiredEventArgs

**Overview**

Event arguments when a port is fired.

**Public Member Functions**

PortFiredEventArgs (string name, NavigationParameters  [page 449] param)

Constructs a new event argument instance.
Properties

<table>
<thead>
<tr>
<th>string</th>
<th>Name [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The name of the source port which fires the event.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NavigationParameters [page 449]</th>
<th>Parameters [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The collection of the navigation parameters passed to the event.</td>
</tr>
</tbody>
</table>

See Also

Common.Controller.Event [page 446]

6.11.2.6 Common.Controller.Navigation Namespace

6.11.2.6.1 Common.Controller.Navigation

Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>NavigationTargetDetails [page 451]</td>
<td>ParamValue [page 451]</td>
</tr>
</tbody>
</table>

6.11.2.6.2 NavigationParameterList

Overview

A list of navigation parameters.

Public Member Functions

<table>
<thead>
<tr>
<th>Dictionary &lt; string, ParamValue [page 451] &gt;</th>
<th>NavigationParameterList (string name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetRow (int index)</td>
<td>Gets a navigation parameter by index.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>AddRow (Dictionary &lt; string, ParamValue [page 451] &gt; row)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a navigation parameter value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>void</th>
<th>Clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clears the navigation parameter list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dictionary &lt; string, ParamValue [page 451] &gt;</th>
<th>AddRow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adds a new empty navigation parameter.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>override string</th>
<th>ToString</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serializes the navigation parameter list as a string.</td>
<td></td>
</tr>
</tbody>
</table>
Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Name [get]</td>
<td>The name of the navigation parameter list.</td>
</tr>
<tr>
<td>int</td>
<td>Rows [get]</td>
<td>The count of the navigation parameters.</td>
</tr>
</tbody>
</table>

See Also


6.11.2.6.3 NavigationParameters

Overview

A flat list of name and value pairs for navigation parameters.

Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NavigationParameters</td>
<td>A constructor for navigation parameters.</td>
</tr>
<tr>
<td>NavigationParameters</td>
<td>A default empty constructor.</td>
</tr>
<tr>
<td>string ResolveSourceTaskUUID</td>
<td>Resolves the source task UUID value and removes it from the parameter list.</td>
</tr>
<tr>
<td>override string ToString</td>
<td>Serializes the current NavigationParameters to a string</td>
</tr>
</tbody>
</table>

Public Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const string SOURCE_TASK_UUID = &quot;$SOURCE_TASK_UUID&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>ForceNewWindow [get, set]</td>
<td>Indicates if the navigation is to appear in a new window.</td>
</tr>
<tr>
<td>bool</td>
<td>AdaptionMode [get, set]</td>
<td>Indicates if the navigation happens in adaptation model.</td>
</tr>
<tr>
<td>bool</td>
<td>OverrideAdaptationMode   [get, set]</td>
<td></td>
</tr>
<tr>
<td>Dictionary&lt; string, ParamValue &gt;</td>
<td>Parameters [get, set]</td>
<td>Returns the collection of parameters.</td>
</tr>
<tr>
<td>string</td>
<td>this [string key] [get, set]</td>
<td></td>
</tr>
<tr>
<td>Dictionary&lt; string, ParamValue &gt;</td>
<td>KeyIndex [get]</td>
<td>Returns the current key index.</td>
</tr>
</tbody>
</table>
### 6.11.2.6.4 NavigationTargetDetails

#### Overview

A flat list of name and value pairs for navigation parameters. For loosely coupled navigation, only flat name and value pairs are supported.

#### Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td><code>CreateLogicalTargetURL</code></td>
<td>Returns the navigation path for the navigation.</td>
</tr>
</tbody>
</table>

#### Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Getter</th>
<th>Setter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td><code>BONameSpace</code></td>
<td>[get, set]</td>
<td></td>
<td>The namespace of the business object.</td>
</tr>
<tr>
<td>string</td>
<td><code>BOName</code></td>
<td>[get, set]</td>
<td></td>
<td>The name of the business object.</td>
</tr>
<tr>
<td>string</td>
<td><code>BONodeName</code></td>
<td>[get, set]</td>
<td></td>
<td>The node name of the business object.</td>
</tr>
<tr>
<td>string</td>
<td><code>BOOperation</code></td>
<td>[get, set]</td>
<td></td>
<td>The operation in the business object.</td>
</tr>
</tbody>
</table>

### See Also


### 6.11.2.6.5 ParamValue

#### Overview

A class for a parameter value.

#### Public Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>override string</td>
<td><code>ParamValue</code></td>
<td>(string value) Initializes with string value.</td>
</tr>
<tr>
<td>override string</td>
<td><code>ToString</code></td>
<td>Serializes to a string.</td>
</tr>
</tbody>
</table>
Properties

| string | Value [get, set] | Gets and sets the parameter value. |

See Also


6.11.2.7 Common.Controls Namespace

6.11.2.7.1 Common.Controls

Classes and Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>IAbstractBaseControl [page 452]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>ILabelControl [page 453]</td>
</tr>
<tr>
<td>Interface</td>
<td>IParameter [page 454]</td>
</tr>
<tr>
<td>Interface</td>
<td>IPlaceableField [page 454]</td>
</tr>
</tbody>
</table>

Enumerations

SAP.BYD.LS.UI.SDK.Common.Controls.ControlUsage

Enumerator:
- Standard
- TransparentTableCellEditor
- TableCellEditor
- ExtendedIdentificationRegion
- FindForm
- DataGridViewCellEditor
- DataGridViewTransparentTableCellEditor
- SelectOptionFindForm

6.11.2.7.2 IAbstractBaseControl

Overview

An abstract base class for all framework controls.
The following chart displays the inheritance diagram for IAbstractBaseControl.
Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>IsVisible [get] Specifies whether the control is currently visible.</td>
</tr>
</tbody>
</table>

Events

<table>
<thead>
<tr>
<th>EventHandler &lt; EventArgs &gt;</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>VisibilityChanged</td>
<td></td>
</tr>
<tr>
<td>SizeChanged</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Common.Controls [page 452]
ILabelControl [page 453]
SDKCustomControl [page 435]

6.11.2.7.3 ILabelControl

Overview

An interface for a label control.
The following chart displays the inheritance diagram for ILabelControl.

The inheriting object points to its source.

See Also

Common.Controls [page 452]
IAbstractBaseControl [page 452]
6.11.2.7.4 IParameter

**Overview**
An interface for the configuration parameter.

**Properties**

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Name [get]</td>
<td>The name of parameter.</td>
</tr>
<tr>
<td>string</td>
<td>Value [get]</td>
<td>The value of parameter.</td>
</tr>
</tbody>
</table>

**See Also**
Common.Controls [page 452]

6.11.2.7.5 IPlaceableField

**Overview**
An interface that represents a placeable field and provides access to the field label and field control. After creating the field instance, use the terminate method to properly cleanup any control instances.

**Public Member Functions**

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>Terminate</td>
</tr>
</tbody>
</table>

**Properties**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAbstractBaseControl [page 452]</td>
<td>Control [get]</td>
<td>The abstract control of the placeable field.</td>
</tr>
<tr>
<td>ILabelControl</td>
<td>LabelControl [get]</td>
<td>The label of the placeable field.</td>
</tr>
</tbody>
</table>

**See Also**
Common.Controls [page 452]
6.11.2.8 Controller Namespace

6.11.2.8.1 Controller

Classes and Interfaces

| Interface | ISDKComponentController [page 455] |

6.11.2.8.2 ISDKComponentController

Overview

Component controller is the runtime access API to the running instance of a component. The main access is exposed through the data container, the available event handlers of the model, the model definition itself, the components textpool and the possibility to call selected controller operations.

Public Member Functions

| void | FireEventHandler (string eventName) |
|      | Calls an event handler with the given name. |
| void | FireOutPlug (string plug, NavigationParameters [page 450] param) |
| void | OpenLink (string url) |
|      | Opens a link with the given URL. |
| void | SynchronizeBackend |
|      | Calls a flush of the datacontainer to the backend |

Properties

| string | ComponentName [get] |
|        | the component name of the designtime model of the actual component controller. |
| ISDKDataContainer [page 459] | DataContainer [get] |
|        | Access to the data container which is the runtime instance of the designtime data model. |
| string | DynamicHelpId [get] |
|        | A component can have a static help ID defined in the model though also a dynamic changing help ID. |
| IEventHandlers [page 448] | EventHandlers [get] |
|        | Access to the collection of defined event-handlers in the model. |
| string | HelpId [get] |
|        | The static defined help ID. |
| ISDKTextPool [page 471] | TextPool [get] |
|        | Access to the textpool of the actual component, e.g. to retrieve language-dependant texts. |
| UXComponent | ComponentDef [get] |
|        | Access to the component model (designtime) of the component. |
Events

<table>
<thead>
<tr>
<th>Event Handler</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventHandler&lt;EventArgs&gt;</td>
<td>DynamicHelpIdChanged</td>
<td>Raised if the dynamic help ID is changed.</td>
</tr>
<tr>
<td>EventHandler&lt;EventHandlerFiredEventArgs&gt;</td>
<td>EventHandlerFired</td>
<td>Raised if an event handler is fired/executed.</td>
</tr>
<tr>
<td>EventHandler&lt;PortFiredEventArgs&gt;</td>
<td>InPortFired</td>
<td>Raised if an inport of the component is fired/triggered.</td>
</tr>
</tbody>
</table>

See Also

Controller [page 455]

6.11.2.9 Controller.Data Namespace

6.11.2.9.1 Controller.Data

Classes and Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISDKBindingScope [page 456]</td>
<td></td>
</tr>
<tr>
<td>ISDKBindingScopeItem [page 457]</td>
<td></td>
</tr>
<tr>
<td>ISDKCodeList [page 457]</td>
<td></td>
</tr>
<tr>
<td>ISDKDataContainer [page 459]</td>
<td></td>
</tr>
<tr>
<td>ISDKDataField [page 460]</td>
<td></td>
</tr>
<tr>
<td>ISDKDataRecord [page 461]</td>
<td></td>
</tr>
<tr>
<td>ISDKDataRecordList [page 462]</td>
<td></td>
</tr>
<tr>
<td>ISDKDataRecordListRow [page 464]</td>
<td></td>
</tr>
</tbody>
</table>

Functions

<table>
<thead>
<tr>
<th>Delegate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Void</td>
<td>RowCreated(ISDKDataRecordListRow row)</td>
</tr>
</tbody>
</table>

6.11.2.9.2 ISDKBindingScope

Overview

An interface for the binding scope that indicates the effective binding path of an element. It is required as a context object to pass to create controls in the context of an effective binding, such as a list or a nested list.
Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Count [get]</td>
<td></td>
</tr>
<tr>
<td>string</td>
<td>EffectiveExpression [get]</td>
<td>The effective expression is the full binding path of the binding scope.</td>
</tr>
<tr>
<td>bool</td>
<td>IsEmpty [get]</td>
<td></td>
</tr>
<tr>
<td>ISDKDataRecordListRow [page 464]</td>
<td>ItemRow [get]</td>
<td>Points to the instance of the data record list row of the binding scope.</td>
</tr>
<tr>
<td>string</td>
<td>ItemRowIdentifier [get]</td>
<td>Gets the item row identifier of the binding scope.</td>
</tr>
</tbody>
</table>

Events

<table>
<thead>
<tr>
<th>Event Handler</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ContextChanged</td>
<td>Raised if the binding scope is changed.</td>
</tr>
</tbody>
</table>

See Also

Controller.Data  [page 456]

6.11.2.9.3 ISDKBindingScopeItem

Overview

An interface for a binding scope.

A binding scope can consist of multiple binding scope items, whereas each binding scope item relates to a list and nested list in the binding scope.

Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>EffectiveExpression [get]</td>
<td>The effective expression is the full binding path of the binding scope item.</td>
</tr>
<tr>
<td>string</td>
<td>RowIdentifier [get]</td>
<td>Gets the item row identifier of the binding scope item.</td>
</tr>
<tr>
<td>string</td>
<td>ScopeExpression [get]</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Controller.Data  [page 456]

6.11.2.9.4 ISDKCodeList

Overview

A code list object grants access to the items of a code list by code, value, and code value.
Public Member Functions

| ObservableCollection<ICodeListItem [page 445]> | GetItems (ValueHelpSortOrderType sorting, CodePresentationModeType hint, string scope) | Returns an observable collection of the items of the code list.  
- Parameters  
  - Sorting: Defines how items are to be sorted.  
  - Hint: Defines how the value is to be formatted (Code only, Value only or CodeAndValue)  
  - Scope: For standard code lists, null. For dynamic code lists, the scope is the data field binding to which the code list belongs. |
|---|---|---|
| bool | TryGetItem (string key, out ICodeListItem [page 445] item) | Gets a code list item by key.  
- Parameters  
  - Key  
  - Item: Returns the ICodeListItem  
Return true if the item can be found in the code list items. |
| bool | TryGetItemByCode (string code, out ICodeListItem [page 445] item, bool ignoring case) | Gets a code list item by code.  
- Parameters  
  - Code: The code which identifies the ICodeListItem.  
  - Item: Returns the ICodeListItem.  
  - Ignoring Case: Indicates whether the search is to be case-sensitive.  
Return true if the item can be found in the code list items. |
| bool | TryGetItemByCodeValue (string value, out ICodeListItem [page 445] item) | Gets a code list item by code value.  
- Parameters  
  - Value: The code value which identifies the ICodeListItem.  
  - Item: Returns the ICodeListItem.  
Return true if the item can be found in the code list items. |
| bool | TryGetItemEx (string key, out ICodeListItem [page 445] item) | Gets a code list item by key.  
- Parameters  
  - Key  
  - Item: Returns the ICodeListItem  
Return true if the item can be found in the code list items. |

Properties

<table>
<thead>
<tr>
<th>int</th>
<th>ItemCount [get]</th>
<th>Gets the count of items in the code list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodeListState</td>
<td>State [get]</td>
<td>Gets the state of the code list</td>
</tr>
</tbody>
</table>
Events

```
EventHandler
<CodeListLoadingEventArgs>

CodeListLoading
Raised once the code list is loaded, for example, for dynamic code lists based on the instance data.
```

See Also

Controller.Data [page 456]

6.11.2.9.5 ISDKDataContainer

Overview

An interface for the data container of the controller, which holds all the runtime data instances according to the defined data model of the current UI component/controller.

Public Member Functions

- **ISDKBoundProperty** [page 465]
  - **BindProperty** (string bindingExpression)
    Allocates a bound property instance, which can be used to monitor data along a given binding path with implicit handling of lead selections of involved lists.

- **IBaseDataElement** [page 444]
  - **GetDataElement** (string path)
    Retrieves a data element by path.

- **string**
  - **ResolveFormattedValue** (string bindingExpression)
    Resolves a bound data field and returns its formatted value.

Properties

- **bool**
  - **IsContextDirty** [get]
    Indicates if work protect mode relevant changes are pending and not saved to the backend.

- **ISDKDataRecord** [page 461]
  - **RootStructure** [get]
    Returns the root data structure or null if no data context has been defined for the component.

Events

```
EventHandler

BackendUpdateFinished
Update of data container from backend finished.
```

See Also

Controller.Data [page 456]
6.11.2.9.6 ISDKDataField

Overview
An interface for a local data field. It holds a single value for the local controller's context.
The data field has an internal string representation of the value which corresponds to the format that is expected by the backend:
- String
  Alphanumeric character string
- Integer
  Numeric character string
- Float
  Numeric character string with '.' character as decimal separator

In addition, it maintains a formatted value which is a string representation of the value in the end user’s default format.
The following chart displays the inheritance diagram for ISDKDataField.

<table>
<thead>
<tr>
<th>Public Member Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>void ProcessRoundtrip (bool force)</td>
</tr>
<tr>
<td>void ProcessRoundtrip</td>
</tr>
<tr>
<td>Roundtrip only happens if the field value has changed.</td>
</tr>
<tr>
<td>bool IsCodelistDefined</td>
</tr>
<tr>
<td>Specifies whether code list is defined.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>string Name [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td>The name of the field.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UXCCTSTypes CCTSType [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td>The CCTS type of the field, which is also known by the modeled data field of the UI component.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool BooleanValue [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value interpreted as a boolean.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DateTime DateValue [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value formatted as a date.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DateTime TimeValue [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value as a time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int IntValue [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value interpreted as an integer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Decimal FloatValue [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value formatted as a float.</td>
</tr>
</tbody>
</table>
string **FormattedValue** [get]
The formatted value based on locale settings, for example.

bool **IsRoundTripPending** [get]
Specifies whether a roundtrip is pending for the field.

string **Value** [get, set]
The "raw" value (internal representation of the type). Controls which are binding against a data field should use the formatted value instead.

bool **IsReadOnly** [get]
Specifies whether the control is read only (metadata).

bool **IsEnabled** [get]
Specifies whether the control is enabled (metadata).

bool **IsMandatory** [get]
Specifies whether input is mandatory (metadata).

**ICodelistItem**

**CodeListItem** [get]
Returns the code list item details.

### Events

**EventHandler** < **ISDKValueChangedEventArgs** [page 469] >

**ValueChanged**
Called when a value has changed.

**EventHandler** < **ISDKStatesChangedEventArgs** [page 469] >

**StatesChanged**
Called when read only, enabled, and mandatory state has changed.

**EventHandler** < **ISDKStatesChangedEventArgs** [page 469] >

**RoundTripPending**
Called when the field has been added to a backend change list.

**EventHandler** < **ISDKValueChangedEventArgs** [page 469] >

**BindingInvalidated**
Called when binding is invalidated due to lead selection change.

### See Also

- **Controller.Data** [page 456]
- **IBaseDataElement** [page 444]

### 6.11.2.9.7 ISDKDataRecord

#### Overview

An interface for a data record.

A data record is a collection of data container elements (**IBaseDataElement**). The root structure of the data container is a data record as well as a data record list row.

The following chart displays the inheritance diagram for **ISDKDataRecord**.
Public Member Functions

<table>
<thead>
<tr>
<th>IBaseDataElement [page 444]</th>
<th>GetMember (string name)</th>
<th>Gets the IBaseDataElement, such as a data record, data record list, or data field.</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>SetMemberValue (string name, string value)</td>
<td>Sets the value of a direct member data field.</td>
</tr>
<tr>
<td>void</td>
<td>SetMemberValue (string name, DateTime value)</td>
<td>Sets the date time value of a direct member.</td>
</tr>
<tr>
<td>IBaseDataElement [page 444]</td>
<td>GetDeepMember (string path)</td>
<td>Gets the element from a deeply stacked structure, for example, a list pane variant, structure. The value of IBaseDataElement may be an empty string if not loaded properly.</td>
</tr>
<tr>
<td>IBaseDataElement [page 444]</td>
<td>GetElements</td>
<td>Returns an array of all elements within the structure.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Name</td>
<td>The name of the data record.</td>
</tr>
<tr>
<td>string</td>
<td>Locator</td>
<td>The locator or binding path of the data record.</td>
</tr>
<tr>
<td>UXCCTSTypes</td>
<td>CCTSType</td>
<td>The CCTS type of the record in case of amount, quantity, measure; otherwise, &quot;none&quot;.</td>
</tr>
</tbody>
</table>

Events

<table>
<thead>
<tr>
<th>Event Handler</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventHandler &lt; ISDKDataRecordEventArgs &gt;</td>
<td>ChildrenChanged Called when one or more children below the IDataRecord have changed.</td>
</tr>
</tbody>
</table>

See Also

Controller.Data [page 456]
IBaseDataElement [page 444]
ISDKDataRecordListRow [page 464]

6.11.2.9.8 ISDKDataRecordList

Overview

An interface for a data record list, which represents a list of data rows. A data record list can be a child of a data structure and can contain all base data elements as children.
The following chart displays the inheritance diagram for ISDKDataRecordList.

![Inheritance Diagram]

The inheriting object points to its source.

**Public Member Functions**

<table>
<thead>
<tr>
<th>ISDKDataRecordListRow [page 464]</th>
<th>Get (int index)</th>
<th>Returns list row by index.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISDKDataRecordListRow [page 464]</td>
<td>Get (string row identifier)</td>
<td>Returns list row by row identifier.</td>
</tr>
</tbody>
</table>

**Properties**

<table>
<thead>
<tr>
<th>String</th>
<th>Name [get]</th>
<th>The name of data record list (listDef).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int</td>
<td>Count [get]</td>
<td>Gets the count.</td>
</tr>
<tr>
<td>Int</td>
<td>LeadSelectedIndex [get, set]</td>
<td>Gets or sets the lead selected index.</td>
</tr>
<tr>
<td>String</td>
<td>LeadSelectedRowIdentifier [get, set]</td>
<td>Sets the lead selected row by its identifier. We recommend using lead selection setter by index; setting lead selection by row identifier is O(n) operation.</td>
</tr>
<tr>
<td>Bool</td>
<td>RoundTripPending [get]</td>
<td></td>
</tr>
<tr>
<td>ISDKDataRecordEventArgs [page 467]</td>
<td>LeadSelectedRow [get]</td>
<td>Returns lead selected row.</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>EventHandler &lt; ISDKDataRecordListEventArgs &gt;</th>
<th>LeadSelectionChanged</th>
<th>Called when the lead selection has changed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventHandler &lt; ISDKDataRecordListEventArgs &gt;</td>
<td>BindingInvalidated</td>
<td>Called when the binding is invalidated.</td>
</tr>
<tr>
<td>EventHandler &lt; ISDKDataRecordListEventArgs &gt;</td>
<td>ListBackendUpdated</td>
<td>Called when the list has been updated by a backend controller operation.</td>
</tr>
<tr>
<td>EventHandler &lt; ISDKDataRecordListEventArgs &gt;</td>
<td>ListCleared</td>
<td>Called when the list is cleared.</td>
</tr>
<tr>
<td>EventHandler &lt; ISDKDataRecordListEventArgs &gt;</td>
<td>ChildrenChanged</td>
<td>Called when a child in the list has changed (called by DataRecordListRow, which itself can be called by DataField).</td>
</tr>
<tr>
<td>EventHandler &lt; ISDKDataRecordListRowChangeEventArgs [page 468] &gt;</td>
<td>RowAdded</td>
<td>Called when the row with the specified index has been added.</td>
</tr>
<tr>
<td>EventHandler &lt; ISDKDataRecordListRowChangeEventArgs [page 468] &gt;</td>
<td>RowRemoved</td>
<td>Called when the row with the specified index has been removed.</td>
</tr>
</tbody>
</table>
**EventHandler < ISDKDataRecordListEventArgs >**

- **StateChanged**
  Called when list state has changed (for example, create enabled state).

**EventHandler < ISDKListRoundtripPendingEventArgs [page 468] >**

- **RoundtripPendingStateChanged**
  Called when the roundtrip pending state has changed.

### See Also

Controller.Data  [page 456]

IBaseDataElement  [page 444]

## 6.11.2.9.9 ISDKDataRecordListRow

### Overview

An interface for a data record list row which represent a data record or structure of one row of a data record list. The following chart displays the inheritance diagram for **ISDKDataRecordListRow**.

The inheriting object points to its source.

### Properties

<table>
<thead>
<tr>
<th>string</th>
<th><strong>RowIdentifier</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The row identifier of the parent data record list.</td>
</tr>
</tbody>
</table>

### See Also

Controller.Data  [page 456]

ISDKDataRecord  [page 461]

ISDKDataRecordListRow  [page 464]
6.11.2.10 Controller.Property Namespace

6.11.2.10.1 Controller.Property

Classes and Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>ISDKBoundProperty [page 465]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class ISDKDataRecordEventArgs</td>
<td>page 467</td>
</tr>
<tr>
<td>Class ISDKDataRecordListEventArgs</td>
<td>page 467</td>
</tr>
<tr>
<td>Class ISDKDataRecordListRowChangeEventArgs</td>
<td>page 468</td>
</tr>
<tr>
<td>Class ISDKListRoundtripPendingEventArgs</td>
<td>page 468</td>
</tr>
<tr>
<td>Class ISDKStatesChangedEventArgs</td>
<td>page 469</td>
</tr>
<tr>
<td>Class ISDKValueChangedEventArgs</td>
<td>page 469</td>
</tr>
</tbody>
</table>

6.11.2.10.2 ISDKBoundProperty

Overview

An interface for a bound property.

A bound property is an observer of a data-value for a given binding path, for example, /Root/List/ListDetails/DataField1. It handles all lead selection and binding changes along the binding path and registers and deregisters itself to and from all necessary events. The exposed properties and events are very similar to the data field properties and events since the bound property is a kind of proxy for the data field though only in a given dynamic context.

Public Member Functions

| void ReBind (ISDKBindingScope [page 456] bindingScope, BindingStyles bindingStyle) | Rebinds the property. |
| void InitializeBinding | Initiates the binding to the data context. This variant starts from the root of the data container. |
| void Terminate | Needs to be called when not used anymore to cleanup and deregister all events |
| void LogPublicEventSuscriptions (string caller) | Writes all event subscription to the application log. Can be used to identify potential memory leaks. |
| ValueFormattingResult [page 445] ValidateAndSetValue (string value) | Sets and validates a value. |
### Properties

<table>
<thead>
<tr>
<th>DataFieldTypes</th>
<th><strong>Field Type</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The basic field type of the connected data field observed by the bound property. Basic field types are, for example, string, integer, float, boolean, and datetime.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UXCCTSTypes</th>
<th><strong>CCTS Type</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The CCTS type of the field of the connected data field observed by the bound property. This is known also from the component model at design time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th><strong>EffectiveBindingExpression</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The effective binding expression of the field.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>BindingValid</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns whether or not the binding is valid.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th><strong>Value</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The raw value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th><strong>FormattedValue</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The formatted value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>BooleanValue</strong> [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value as a boolean.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>decimal</th>
<th><strong>FloatValue</strong> [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value as a float.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>int</th>
<th><strong>IntValue</strong> [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value as an integer.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DateTime</th>
<th><strong>DateValue</strong> [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value as a date.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DateTime</th>
<th><strong>TimeValue</strong> [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value as a time.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>string</th>
<th><strong>StringValue</strong> [get, set]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Value as a string.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ISDKCodeList [page 457]</th>
<th><strong>CodeList</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The code list (if any).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ICodeListITem [page 445]</th>
<th><strong>CodeListItem</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The current selected code list item (if any)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsReadOnly</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies whether the metadata indicates a read only state.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsEnabled</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies whether the metadata indicates an enabled state.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsMandatory</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies whether the metadata indicates a mandatory state.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsMetaDataAvailable</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies whether metadata has been provided for the bound field. “true” unless field binding is used. For field binding, “true” if metadata has been returned by the backend.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsRoundTripPending</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies whether a roundtrip is pending.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsCodeListDefined</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specifies whether a code list is defined.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DependentPropertyType</th>
<th><strong>Definition</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The bound property’s definition from the designtime component model.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsDataFieldBound</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns whether or not a data field is actually bound.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bool</th>
<th><strong>IsRecordBound</strong> [get]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns whether or not a record is bound.</td>
</tr>
<tr>
<td>ISDKDataRecordList [page 462]</td>
<td><strong>BoundRecordList</strong> [get]</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>ISDKDataField [page 460]</td>
<td><strong>BoundDataField</strong> [get]</td>
</tr>
<tr>
<td>ISDKDataRecord [page 461]</td>
<td><strong>BoundDataRecord</strong> [get]</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>EventHandler &lt; ISDKValueChangedEventArgs [page 469] &gt;</th>
<th><strong>ValueChanged</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>EventHander &lt; ISDKStatesChangedEventArgs [page 469] &gt;</td>
<td><strong>StatesChanged</strong> Raised when the bound value is changed (or the binding itself changes).</td>
</tr>
<tr>
<td>EventHander &lt; ValidationFailedEventArgs [page 443] &gt;</td>
<td><strong>ValidationFailed</strong> Raised when the validation has failed.</td>
</tr>
<tr>
<td>EventHander&lt; EventArgs &gt;</td>
<td><strong>ValidationSucceeded</strong> Raised when the validation has succeeded.</td>
</tr>
<tr>
<td>EventHander &lt; ISDKValueChangedEventArgs [page 469] &gt;</td>
<td><strong>MessageAdded</strong> Called when a message has been changed.</td>
</tr>
<tr>
<td>EventHander &lt; ISDKValueChangedEventArgs [page 469] &gt;</td>
<td><strong>MessageRemoved</strong> Raised when the messages have been removed.</td>
</tr>
<tr>
<td>EventHander &lt; ISDKValueChangedEventArgs [page 469] &gt;</td>
<td><strong>RoundTripPending</strong> Raised when the field has been added to a backend changelist.</td>
</tr>
<tr>
<td>EventHander&lt; EventArgs &gt;</td>
<td><strong>ListLeadSelectionChanging</strong> Raised when the lead selection of the list changes.</td>
</tr>
</tbody>
</table>

### See Also

Controller.Property [page 465]

### 6.11.2.10.3 ISDKDataRecordEventArgs

#### Overview

Event arguments that are passed in the `ChildrenChanged` data record event.

#### Public Types

```csharp
enum DataRecordChangeEventsEnum { ChildrenChanged }
```

### See Also

Controller.Property [page 465]
6.11.2.10.4 ISDKDataRecordListRowChangeEventArgs

**Overview**
Event arguments for the change event of a data record list row.

**Properties**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract ISDKDataRecordListRow</td>
<td>[page 464] Row [get] The data record list row that is the source of the event.</td>
</tr>
<tr>
<td>abstract string</td>
<td>RowIdentifier [get] The row identifier for the data record list row that is the source.</td>
</tr>
</tbody>
</table>

**See Also**
Controller.Property [page 465]

6.11.2.10.5 ISDKDataRecordListRowChangeEventArgs

**Overview**
Event arguments for the change event of a data record list row.

**Properties**

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract ISDKDataRecordListRow</td>
<td>[page 464] Row [get] The data record list row that is the source of the event.</td>
</tr>
<tr>
<td>abstract string</td>
<td>RowIdentifier [get] The row identifier for the data record list row that is the source.</td>
</tr>
</tbody>
</table>

**See Also**
Controller.Property [page 465]

6.11.2.10.6 ISDKListRoundtripPendingEventArgs

**Overview**
Event arguments for the ListRoundtripPending event.
Properties

abstract `RoundtripPendingState` `State` [get]
The state that causes the roundtrippending-event (reload, paging, add row, remove row, expand all).

See Also

`Controller.Property` [page 465]

6.11.2.10.7 ISDKStatesChangedEventArgs

Overview

Event arguments for `StatesChanged` event.

Properties

abstract bool `IsEnabled` [get]
Indicates if enabled metadata has changed.

abstract bool `IsMandatory` [get]
Indicates if mandatory metadata has changed.

abstract bool `IsReadOnly` [get]
Indicates if read-only metadata has changed.

See Also

`Controller.Property` [page 465]

6.11.2.10.8 ISDKValueChangedEventArgs

Overview

Event arguments for the `ValueChanged` event of data field.

Public Types

enum `ValueChangedEventsEnum` { ValueChanged, StatesChanged, BindingInvalidated, MessageRemoved, MessageAdded, RoundTripPending, FieldValidated }
The source of the `ValueChanged` event.

Properties

abstract `ISDKDataField` [page 460] `Field` [get]
The reference to the source data field.
6.11.2.11 Controls Namespace

6.11.2.11.1 Controls

Classes and Interfaces

| Interface | ISDKControlFactory [page 470] |

6.11.2.11.2 ISDKControlFactory

Overview

A factory class to create bound controls and placeable fields defined in the model.

Public Member Functions

<table>
<thead>
<tr>
<th>Interface</th>
<th>CreateBoundControl (AbstractControlType FieldDefinition, ISDKBindingScope [page 456] BindingScope, UXCCTSTypes EnclosedCCTSType)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAbstractBaseControl [page 452]</td>
<td>Write and validate properties, and create the control.</td>
</tr>
<tr>
<td>Parameters:</td>
<td></td>
</tr>
<tr>
<td>○ FieldDefinition: The field definition that defines the bound control.</td>
<td></td>
</tr>
<tr>
<td>○ BindingScope: The binding scope if the bound control is to be created in a context of a list or nested list.</td>
<td></td>
</tr>
<tr>
<td>○ EnclosedCCTSType: Neded if is a field inside a complex CCTS field, such as Amount, Quantity or Measure.</td>
<td></td>
</tr>
<tr>
<td>Creates a bound control of the framework for a given field definition. Using CreatePlaceableFields is more convenient since it allows defining all needed properties, calculation rules, and so on, directly at design time arbitrary to any direct UI placement and the instantiation happens here from the factory.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interface</th>
<th>CreatePlaceableField (string FieldName, ISDKBindingScope [page 456] BindingScope)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPlaceableField [page 454]</td>
<td>Write and validate properties, and create the placeable field.</td>
</tr>
<tr>
<td>Parameters:</td>
<td></td>
</tr>
<tr>
<td>○ FieldName: The name of the placeable field defined in the model.</td>
<td></td>
</tr>
<tr>
<td>○ BindingScope: The binding scope if the placeable field is to be created in a context of a list or nested list.</td>
<td></td>
</tr>
<tr>
<td>Creates an instance of the placeable field, which is a framework control with all the intrinsic framework behavior that is modeled in the UI component model.</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Controller.Property [page 465]
6.11.2.12 Model Namespace

6.11.2.12.1 Model

Classes and Interfaces

| Interface   | ISDKTextPool [page 471] |

6.11.2.12.2 ISDKTextPool

Overview

An interface that permits access to the text pool of the component model, for example, to get language dependant texts in the current language.

Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string Format</td>
<td>Gets the text with given values for parameters.</td>
</tr>
<tr>
<td>bool IsCalculated</td>
<td>Indicates if the given text pool entry is based on a calculation rule.</td>
</tr>
<tr>
<td>string Lookup</td>
<td>Returns a static text from the text pool.</td>
</tr>
<tr>
<td>string LookupRaw</td>
<td>Returns the raw text pool entry value, without any dynamic replacement.</td>
</tr>
</tbody>
</table>

See Also

Model [page 471]
6.11.3 Tasks

6.11.3.1 Create a Custom Pane or Custom Control

Overview

You use Microsoft Visual Studio 2010 to create custom panes and custom controls as Microsoft Silverlight components. You use the classes and methods defined by the Application Programming Interface (API) for Software Development Kit (SDK) Custom Panes and Custom Controls to implement your component so that it can be included in the interface of the SAP solution.

Prerequisites

- You have installed and the Microsoft Silverlight Developer Runtime.
- You have downloaded and installed the Microsoft Visual Studio templates for custom panes, ByDCustomPaneTemplateSetup.msi.

Procedure

1. Set Up Your Environment and Create a Project
   a. To create a new project in Microsoft Visual Studio 2010, click File > New > Project. The New Project dialog appears.
   b. Click Visual C# under Installed Templates. The following SAP Business ByDesign custom templates are listed:
      - ByDesign Custom Control
      - ByDesign Custom Pane
   c. Click ByDesign Custom Pane to create a custom pane, or click ByDesign Custom Control to create a custom control.
   d. In the Name area, enter a name and click OK. Two projects are created. The first project contains a template that you can use to create your custom component. The second project contains a debugging environment that you can use to test your component.
   e. In the Solution Explorer, expand the Lib folder and verify that the following DLLs have been added to your project:
      - sapLSUICoreModel1.dll
      - sapLSUISDK.dll
      - sapLSUISDKCommon.dll

2. Implement Your Microsoft Silverlight Component
   a. In the Solution Explorer, expand your project and double-click the C# template file (for example, CustomPane.cs). The file is displayed in the editor.
   b. Use methods from the API for SDK Custom Panes and Custom Controls to define data fields and assign values from the data model.
For example, use the ISDKDataField interface to create a data field:

```csharp
ISDKDataField myStringData =
    this.Controller.DataContainer.RootStructure.GetMember("StringData") as ISDKDataField
```

3. **Test Your Component**
   a. Right-click the debugging environment project and click Properties Silverlight Applications. The project properties are displayed.
   b. If your project does not appear in the Silverlight Applications list, add it to the list.
      1. Click Add. The Add Silverlight Application dialog appears.
      2. Select Use an existing Silverlight project in the solution and select your project.
      3. Select Add a test page that references the control.
      4. Select Enable Silverlight debugging and click Add.
      5. Save the changes to your debugging environment project.
   c. In the Solution Explorer, right-click the debugging environment project and click Debug Start New Instance.

**See Also**

Application Programming Interface (API) for SDK Custom Panes and Custom Controls [page 433]

6.11.3.2 Create an Embedded Component for a Custom Pane

**Overview**

You use an embedded component to provide a container for a custom pane. The embedded component provides bindings that access the data fields of the custom pane, and can be bound to data fields in a business object. You can add an embedded component directly to screens.

**Prerequisites**

- You have a custom pane in the form of a Microsoft Silverlight application (.xap file).
- You have a business object with data fields that correspond to the fields of the custom pane.

**Procedure**

1. In the Solution Explorer, right-click your project and select Add New Item. The Add New Item dialog appears.
2. Select Embedded Component.
3. Enter a Name for the embedded component, and click Add. The embedded component is added to your project.
4. In the Solution Explorer, right-click the embedded component and select Upload Custom Pane. The Add Custom Pane to Embedded Component dialog appears.
5. Click the **Browse** button and navigate to your custom pane (*.xap file).

6. Enter the **Assembly Name** and **Assembly Type** that were used to develop the custom pane, and click **OK**.

7. Double-click the embedded component in the **Solution Explorer**.
   The embedded component is opened in the user interface designer.

8. Create data fields and bindings for the fields in the custom pane.
   a. Click the **Display <-> Edit** button to change to edit mode.
   b. Click the **Embedded Component** tab.
   c. Click the **Data Model** tab.
   d. For each field in the custom pane, add a data field to the data model.
      1. Right-click the **Root** node and select **Add Data Field**.
         A new field is added to the data model.
      2. Right-click the data field, click **Rename**, and enter a name for the field.
         Rename the field to match the name of the data field in the custom pane.
   e. Click the **Controller** tab.
   f. Create a binding for each data field.
      1. Right-click **Bindings**, and click **Add Binding**.
         A new binding is added.
      2. Right-click the binding, click **Rename**, and enter a name for the binding.
         Rename the binding to match the name of the data field in the business object.
      3. Select the binding and the corresponding data field, and click the **Bind** button.
         The **Binding Expression** field displays the path to the bound field in the data model.
   g. Click the **Save and Activate** button.

**See Also**

*Custom Panes and Custom Controls Quick Guide* [page 432]
*Add an Embedded Component to a Screen* [page 474]

### 6.11.3.3 Add an Embedded Component to a Screen

**Overview**

You can add an embedded component to a screen, and then bind the fields of the embedded component to the data fields of a business object.

**Prerequisites**

- You have added your custom pane to an embedded component, and have created bindings to the custom pane’s data elements.
You have a business object that contains data fields that correspond to the bindings in the embedded component.

Procedure

1. In the Solution Explorer, right-click your business object and select Create Screens. The Create Screens dialog opens.
2. Enter a name for your screens and select the screen types to create, and then click OK.
3. In the Solution Explorer, right-click the screen that you want to customize and select Open in UI Designer. The user interface designer (UI designer) opens.
4. Click the Display <-> Edit button to change to edit mode.
5. Add a blank row or column to the screen.
   a. Select a pane on the screen.
   b. Right-click and select either Layout Insert Row or Layout Insert Column. A blank row or column is added next to the pane you selected.
6. In the Configuration Explorer, locate your embedded component.
   a. To view the Configuration Explorer, click View Configuration Explorer.
   b. In the Configuration Explorer, expand the Repository folder.
   c. Expand the folder for your project. The name of this folder is the name of your project followed by _MAIN.
   d. Expand the SRC folder. Your embedded component is displayed in this list.
7. Drag your embedded component from the Configuration Explorer to the blank row or column on the Designer tab. The embedded component replaces the blank row or column.
8. In the Embedded Component area, click Bind. The Embedded component binding dialog appears.
9. Bind each of the fields in the data model to the bindings of the embedded component.
   a. In the Parent Data Model area, select a data field.
   b. In the Controller Interface of Embedded Component area, select the binding for the corresponding field in the embedded component.
   c. Click Bind. The Binding Details area shows the name, binding path, and type of binding.
10. Click OK.
11. Click the Save and Activate button.
12. Close the UI designer.

See Also

Custom Panes and Custom Controls Quick Guide  [page 432]
Create an Embedded Component for a Custom Pane  [page 473]
6.11.4 Example: Best Employee Graph

Overview
In this example you create a graph that displays the assigned tasks and completed tasks for a list of employees. You can drag-and-drop employees onto the graph, and when an employee qualifies as the “best employee”, a new business object instance is created.

To achieve this, you will perform the following tasks:
- Upload a custom pane into an embedded component
- Place the embedded component into a quick activity floorplan (QAF)
- Preview the QAF to display and test the custom pane

Prerequisites
- You are familiar with creating solutions and business objects in the SDK. For more information see:
  - Create a Customer-Specific Solution  [page 152]
  - Business Objects Quick Guide  [page 297].

Procedure
1. Save the custom pane from here to your local system with the file extension .xap. This custom pane was created in Microsoft Visual Studio 2010.
2. In the SDK, do the following:
   a. Create a solution.
   b. Add a business object with the name Best_Employee to your solution.
   c. Add elements to your business object to match the fields in the custom pane.
      ```
      element ID : ID;
      element FirstName : LANGUAGEINDEPENDENT_EXTENDED_Text;
      element LastName : LANGUAGEINDEPENDENT_EXTENDED_Text;
      element AssignedTask : IntegerValue;
      element FinishedTask : IntegerValue;
      ```
   d. Right-click your business object and select Create Screens Floorplan Scenario with Navigation. You can accept the default name for the screens.
   e. Right-click your project and select Add New Item Embedded Component. Enter the name EmbeddedComponent_QAF for the embedded component.
   f. Right-click your embedded component and select Upload Custom Pane.
      1. Click Browse and locate the custom pane .xap file.
      2. For the Assembly Name enter ByDCustomPane.dll and for the Assembly Type enter ByDCustomPane.CustomPaneQAF.
   g. Double-click the embedded component to open it in the UI designer.
3. In the UI designer, do the following:
a. On the DataModel tab, add data fields to match the fields in the custom pane (ID, FirstName, LastName, AssignedTask, and FinishedTask).

b. In the Property Explorer, change the Type property to integer for the AssignedTask and FinishedTask data fields.

c. On the Controller tab, add bindings for each data field (ID, FirstName, LastName, AssignedTask, and FinishedTask). Bind each binding to the corresponding data field.

d. Add an Event Handler with the name WindowsSaveBOEventHandler. Select the Type as WindowAction and the Action Type as Save.
This event handler is called from the custom pane to create a new instance of the Best_Employee business object.
4. In the SDK, double-click the QAF to open the floorplan in the UI designer.

5. In the UI designer, do the following:
   a. Click **Add FlexLayoutRow** to add a row to the screen.
   b. Drag your embedded component from the **Configuration Explorer** to the new row. Your embedded component is located in the folder: `<project_name>_MAIN > SRC`
   c. Click **Bind** to open the **Embedded Component Binding** dialog box.
   d. Bind the fields from the floorplan to the corresponding fields in the embedded component.
   e. Save and activate the embedded component.

   ![Image of UI Designer](image)

   e. Save and activate the QAF.
6. In the SDK, right-click the object work list (OWL) and select Preview Screen. On the OWL, click New to open the QAF. The QAF is displayed, including the custom pane.
   a. Drag an employee from the list on the left of the custom pane onto the graph on the right. The employee’s tasks are visualized on the graph. If this employee has the highest number of finished tasks, the employee is designated as the new “best employee” and a new business object instance is created.
   b. Drag other employees onto the graph.
   c. Close the QAF to return to the OWL.
   d. Click Go to refresh the OWL. The list now includes any employees added from the custom pane.
6.12 Web Services

6.12.1 Web Services Quick Guide

The SDK allows you to create a Web service on an SAP business object or on any of your business objects, which can then be consumed by a client application. This enables remote access to the business object data. You can integrate an external Web service into your solution and address it in your business logic to enable access from the SAP solution to a remote application. You can also select a set of inbound and outbound services to create a communication scenario. This allows customers to create communication arrangements in the SAP solution to exchange business documents electronically. All communication for Web services is based on SOAP.

Business and Technical Background

Web Services

A Web service is a set of programming standards that enables interoperation between different software applications, running on a variety of platforms or frameworks. Web services use Extensible Markup Language (XML), Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL) to integrate software applications in the internet. XML is used to tag the data, SOAP is used to transfer the data, and WSDL is used to describe the Web service. Web service discovery mechanisms such as Universal Description, Discovery and Integration (UDDI), and Web Services Inspection Language (WSIL) can locate the Web services provided by a Web site.

For more information, see Web Services [page 483].

Web Service Operations

Web services contain multiple service operations. These service operations are assigned to service interfaces which serve as collectors for specific operation types. There is one endpoint per service interface.

For more information, see Web Service Operations [page 487].

Communication Scenarios

Communication scenarios are sets of inbound and outbound services that allow customers to create communication arrangements in the SAP solution. Inbound services are Web services that you have created based on SAP business objects or your business objects, or Web services provided by SAP. Outbound services are external Web services that you have integrated into your solution.

For more information about defining a communication scenario, see Define a Communication Scenario [page 503].

Communication Arrangements

Communication arrangements help you to configure the electronic data exchange between the SAP solution and a communication partner. Communication arrangements can be set up for multiple business documents and communication methods.

A communication arrangement is based on a communication scenario, and provides an alternative method to authorize a user to access a business object. In a communication arrangement, you can specify an authentication method to authorize a user to call a Web service that you have created.

For more information, see Communication Arrangements Quick Guide.
Tasks

Create a Web Service
You can create a Web service based on an SAP business object or one of your business objects. You select the business object elements and define the service operations that carry out tasks on the business object, for example, creating or updating the business object data.
For more information, see here [page 494].

Test a Web Service
Before you can use your Web service in a client application, you must test it in a separate testing tool.
For more information, see here [page 496].

Modify a Web Service
You can modify a Web service that you have created based on an SAP business object or one of your business objects. To modify a Web service, you can do the following:
- Add or remove business object elements for the service operations already defined in the Web service.
- Add or remove service operations.
- Adjust the derived names for the message data types used in the WSDL file.
For more information, see here [page 497].

Edit Web Service Authorizations
You can edit the authorization of the business users assigned to work center views to use the Web service operations. Proceed as follows:
1. In the Solution Explorer, double-click the .wsauth file to open it for editing.
2. You have the following options:
   - To add a work center view, click Add and select an existing view, or enter a technical name for a new view. For more information about the process of creating a new view for use with a Web service, see Create a Web Service [page 494].
   - To remove authorization from a work center view, select the line in the table and click Remove. Note that this does not delete any work center views created specifically for the Web service.
   - To assign the Web service operations to work center views, select a view and then select or deselect the checkboxes in the lower table.
   - To find out where a work center view is stored in the UI Designer, view the Repository Path column.
3. Save your changes and activate the Web service.

Integrate an External Web Service
You can integrate an external Web service into your solution by importing the WSDL file of this Web service. You can then address the Web service in your business logic to enable your solution to access a remote application using SOAP-based communication.
For more information, see here [page 499].
Modify an External Web Service Integration

You can modify an external Web service integration that you have created to access a remote application using SOAP-based communication. You can modify an external Web service integration as follows:

- Replace the existing WSDL file with a new file.
- Add or remove XML schema documents, if any are referenced in the WSDL file.

For more information, see here [page 501].

Define a Communication Scenario

In a communication scenario, you can define a set of inbound and outbound services for a business process. Based on a communication scenario, a customer can create a communication arrangement in the SAP solution to exchange business documents electronically.

For more information, see here [page 503].

Modify a Communication Scenario Definition

You can modify a communication scenario that you have created based on a business-to-business or application integration communication type. You can modify a communication scenario as follows:

- Change the communication type on which the communication scenario is based.
- Add or remove inbound and outbound services.

For more information, see here [page 504].

Test a Service Operation

You can test the execution of service operations provided by an external Web service. Proceed as follows:

1. In the Solution Explorer, double-click the .wsid file of the external Web service integration for which you want to test the execution of service operations.
2. Select the Test Tool tab and select a service operation.
3. Provide test values for the request message in the Request tab and click Submit Request.
4. Enter the name of the communication scenario that contains the external Web service integration as an outbound service.
5. If the communication scenario is based on a business-to-business communication type, select the business partner ID.
6. Click OK.
7. You can see the response message in the Response tab and verify if it is correct.

- The Test Tool tab is only available after you have activated the .wsid file.
- The Test Tool tab is not available when the following apply:
  - You cleaned the .wsid file.
  - You deleted the WSDL file.
  - You uploaded a new WSDL file to replace the original file.

Define the Business Logic to Call a Service Operation

If you have integrated an external Web service into your solution, you typically call the service operations provided by the Web service in a script file. The external Web service integration generates a library in the project namespace.
This library is similar to a reuse library and you can use it to implement your business logic to call the service operations.

For more information, see here [page 505].

**Look for Deprecated A2X Web Services in the Solution**

1. In the Application and User Management work center, select the Service Explorer view.
2. Choose Advanced to open the advanced search pane.
3. Under Operation in the field Operation Release Status, select Deprecated and in the field Operation Deprecation Release, specify the release in which you want to search for deprecated Web services. Choose Go.

The system displays the list of deprecated A2X Web services in the release you specified.

6.12.2  Business and Technical Background

6.12.2.1  Web Services

**Overview**

A Web service is a set of programming standards that enables interoperation between different software applications, running on a variety of platforms or frameworks. Web services use Extensible Markup Language (XML), Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL) to integrate software applications in the internet. XML is used to tag the data, SOAP is used to transfer the data, and WSDL is used to describe the Web service. Web service discovery mechanisms such as Universal Description, Discovery and Integration (UDDI), and Web Services Inspection Language (WSIL) can locate the Web services provided by a Web site.

In the SAP solution, internal communication between business objects is implemented as service integration. For more information, see the Service Integration Quick Guide [page 414].

**Features**

**Web Service Operations**

The function in the Web service is defined in service operations performed on specified elements in the selected business object.

For more information, see Web Service Operations [page 487].

**Authorization Concept**

You can authorize users to call a Web service that you have created as follows:

- **Web Service Authorization Object**
  
  You can define a Web service authorization object for the service operations in the Web service. The authorization concept is illustrated in the following diagram:
Web Service Authorization Process

The Web service authorization object is assigned to work center views. For every assigned work center view, an external application containing the selected Web service operations is created. All users who have these work center views assigned to them are subsequently authorized to call the Web service operations. Any restrictions, such as read-only access to data, that have been defined for the assigned work center view are also taken into account when accessing business objects using the service operation.

For more information about defining Web service authorizations, see Create a Web Service [page 494].

- Communication Arrangement
  You can add a Web service that you have created as an inbound service to a communication scenario. Based on this communication scenario, you can create a communication arrangement in the SAP solution. In the communication arrangement, you can provide an authentication method to authorize a user to call the Web service.

  For more information about communication arrangements, see Communication Arrangements Quick Guide.

  If you use a communication arrangement to authorize a Web service that has a Web service authorization object, you can delete the Web service authorization object.

Testing Consumption and Error Handling

To test the Web service, you download the WSDL file, import it into your client testing tool, and then assign the relevant work center views to a test user. For more information, see Test a Web Service [page 496].

The following list is intended to help you test the Web service:

- Checking Endpoint Activation
  Every SDK tenant supports the WSIL standard. The WSIL service is a web site with the URL https://[hostname]/sap/bc/srt/wsil that lists the active Web services and endpoints provided by SAP. Enter the URL for the WSIL service in your browser to check whether the endpoint is active and accessible.

- XML Validation
  As for all inbound service interfaces, the Web service runtime only performs a minimal set of checks (for example, checking maximum string lengths) and not general or strict validation of the inbound message. Superfluous tags are simply ignored.

- Fault Messages
  If an error occurs within a Web service, the Log element contains the details of the error in the response message. For more information about Log element, see Web Service Operations [page 487].

Web Services Description Language

The Web Services Description Language (WSDL) is an XML-based language that is used for describing the functions offered by a Web service. A WSDL file provides a description of how the Web service can be called, which parameters it expects, and which data structures it returns. WSDL is often used in combination with SOAP and an XML Schema document to define Web services in the internet. A client application connecting to a Web service can read the WSDL
file to determine which service operations are available on the server. The client application can then use SOAP to call one of the service operations listed in the WSDL file. Some of the data types used are defined in a separate XML Schema document that is referenced in the WSDL file. For more information, see XML Schema Document.

In a WSDL file the following elements are used for the definition of Web services:

- **Types**
  A container for data type definitions such as XML Schema definition (XSD).

- **Message**
  An abstract, typed definition of the data being communicated.

- **Operation**
  An abstract description of an action supported by the service.

- **Port Type**
  An abstract set of operations supported by one or more endpoints.

- **Binding**
  A concrete protocol and data format specification for a particular port type.

- **Port**
  A single endpoint defined as a combination of a binding and a network address. Optionally, a binding can be defined between the operation name and the SOAP action.

- **Service**
  A collection of related endpoints.

A WSDL file defines Web services as collections of network endpoints, or ports. A port is defined by associating a network address with a reusable binding. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: messages and port types.

The following is an example of a basic WSDL file that refers to an XML Schema document:
XML elements and attributes are usually placed inside a container called a namespace. Inside the `<types>` element, the `targetNamespace` attribute is used if you want to place elements and attributes from the default namespace that is defined using the `xmlns` attribute into a different namespace. The value of the `schemaLocation` attribute is the location of the XML Schema document to use for the target namespace. Usually this value can be a URL or the location of the XML Schema document relative to the location of the WSDL file on your local computer. In the above example, the WSDL file refers an XML Schema document `PayrollControlSchema.xsd` that is located in the same folder as the WSDL file.

The SDK only accepts the relative location of the XML Schema document for the value of the `schemaLocation` attribute.

**XML Schema Document**


Some of the data types used in a WSDL file are embedded in the file in the form of an XML Schema document. This modular schema design maximizes the potential for reusability, flexibility, and maintainability of Web services components. As an XML Schema document is written in XML, there is no need for a parser. An XML Schema document defines a rich set of data types, for example, numeric data types, string data types, and date data types, and also makes it easier to validate documents based on namespaces.
An XML Schema defines the following:

- Elements and attributes that can appear in a document.
- Whether an element is a child element.
- The order and the number of child elements.
- Whether an element is empty or can include text.
- Data types for elements and attributes.
- Default and fixed values for elements and attributes.

The following is an example of a basic XML Schema document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    targetNamespace="http://example.com/schema/"
>
    <xsd:import namespace="http://example.com/schema/common"
        schemaLocation="PayrollCommonSchema.xsd"/>

    <xsd:element name="PayrollGroupAnnualSchedule" nillable="false"
        type="pcns:PayrollGroupPayrollScheduleType"/>

    <xsd:complexType name="PayrollGroupPayrollScheduleType">
        <xsd:annotation>
            <xsd:documentation>Denotes the list of payroll schedules for the calendar year for the payroll group</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="PayrollGroupCode" nillable="false"
                type="cns:PayrollGroupCode"/>
            <xsd:element maxOccurs="unbounded" name="PayrollSchedule"
                nillable="false" type="pcns:PayrollScheduleType"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

In the above example, the XML Schema document refers another XML Schema document PayrollCommonSchema.xsd.

See Also

Web Services Quick Guide [page 480]

6.12.2.2 Web Service Operations

Overview

Web services contain multiple service operations. These service operations are assigned to service interfaces which serve as collectors for specific operation types. There is one endpoint per service interface.
Core Concepts

Statelessness

By definition and design, Web service operations are stateless synchronous inbound operations. This means that no resources such as enqueue locks or buffers are shared or maintained across subsequent calls. To ensure reasonable transaction sizes for mass operations, it is recommended that you avoid excessively large transactions because the time required to transfer huge amounts of data across the network may cause communication time-outs.

Action Code

The action code is a coded representation of an instruction to the recipient of a message telling the recipient how to process it.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Create</td>
<td>The element is to be created at the recipient. The element must not exist at the recipient. The element ID and all data must be transferred.</td>
</tr>
<tr>
<td>02</td>
<td>Change</td>
<td>The element is to be changed at the recipient. The element must exist at the recipient. The element ID and all data must be transferred.</td>
</tr>
<tr>
<td>03</td>
<td>Delete</td>
<td>The element is to be deleted at the recipient. The element must exist at the recipient. The element ID must be transferred. No data should be transferred, with the exception of elements that are mandatory due to their cardinality.</td>
</tr>
</tbody>
</table>
| 04   | Save    | The element is to be saved at the recipient. The element can exist at the recipient. If, at the recipient, the element:  
- Already exists, it is changed  
- Does not exist, it is created  
The element ID and all data must be transferred. |
| 05   | Remove  | The element is to be deleted at the recipient. The element can exist at the recipient. If, at the recipient, the element:  
- Already exists, it is changed  
- Does not exist, it is created |
| 06   | No action| No action is to be carried out for the element at the recipient. The element must exist at the recipient. The element ID and all data must be transferred. |

Complete Transmission Indicator

Due to the synchronous nature of Web services, the selective transmission of data during updates is essential for performance. In addition it cannot be assumed that the whole business object instance has to be read before an update takes place. This is especially true in cases where new field nodes are added via extensions. Without selective transmission, the new fields would be cleared and the new node instances would be deleted when an update message is processed. The CompleteTransmissionIndicator is used to indicate if the data in the requested operation represents all the data of an element or only part of it.

For operation message data types, the following rules apply:

- The CompleteTransmissionIndicator attribute is only used for lists, not for elements.
- For all updates, the default value is false, that is, by default, lists are always incomplete.
- For updates, if the value is true, all instances of the associated business object node that are not part of the message will be deleted.
• For creations or deletions, the indicator is assumed to be true or you can set it yourself explicitly to true.

Log

The Log element represents the result of an operation as a series of requests and responses. The most important parts of the Log element are:

Description of Key Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaximumLogItemSeverityCode, LogItemSeverity</td>
<td>These elements provide you with information on the degree of success of a given operation. The possible values are:</td>
</tr>
<tr>
<td></td>
<td>• 1 Information</td>
</tr>
<tr>
<td></td>
<td>• 2 Warning</td>
</tr>
<tr>
<td></td>
<td>• 3 Error</td>
</tr>
<tr>
<td>LogItem</td>
<td>The aggregated data type for the LogItem contains the following elements:</td>
</tr>
<tr>
<td></td>
<td>• In a response, the ReferenceObjectNodeSenderTechnicalId references the ObjectNodeSenderTechnicalId used in the request message. For more information, see the Object Node Sender Technical ID section below.</td>
</tr>
<tr>
<td></td>
<td>• ReferenceMessageElementName</td>
</tr>
<tr>
<td></td>
<td>This value denotes the element in the request that the log item refers to. It enables you to relate error messages to lines and columns in the query or request message, irrespective of the ObjectNodeSenderTechnicalId.</td>
</tr>
</tbody>
</table>

Object Node Sender Technical ID

When a request contains at least one element that occurs more than once, it must be possible to identify the corresponding elements in the response and the LogItem. To this end, the ObjectNodeSenderTechnicalId element is contained on every level of the request message as an optional element. The corresponding element ReferenceObjectNodeSenderTechnicalId is contained in the LogItem element at the root level of the response message.

The ObjectNodeSenderTechnicalId is also used to identify a failed newly created instance in a mass operation. Because the instance was not created in the database in these cases, no UUID or ID is returned and only the ReferenceObjectNodeSenderTechnicalId indicates which part of the request has failed. The ObjectNodeSenderTechnicalId contains transient values that establish the correspondence between elements only for a single call.

Change State Identifier

Due to the stateless behavior of the Web service operations, no transactional or logical locking is possible. However, the ChangeStateId provides a way to ensure that a modifying operation is not executed if the state of the underlying business object instance in the database has changed since the last time that data was read. The ChangeStateId is an uninterpretable string that is provided by all operations and may be consumed by all modifying operations.

Update Operations

If the ChangeStateId is provided when calling an update operation, the system does not perform the operation if the state of the business object instance in the database has changed since the ChangeStateId was computed.

Create Operations

For create operations, there is no previous state of the underlying business object instance. Therefore the change state identifier is not modeled in single instance create operations and is optional in mass instance operations that support the corresponding action codes for creation.
If the change state identifier is specified and the specified value for a business object instance is different from the actual value computed from the database, an error message with the value "SBM.SBM.CSM" (change state mismatch) in the element LogItem.LogItemCategoryCode is returned for that business object instance in the Log element of the response.

| Operations which create or update data, return the new IDs and the ChangeStateId only for those business object instances for which the operation succeeded. |

### Read Operation

The read operation returns detailed information of one or more instances of a business object. The part of the business object returned can be limited by the defined business object view. In general, the read operation always forms a pair with the corresponding create or update operation. Therefore, if a full or partial update operation for a business object exists, a full or partial read operation must also exist in order to retrieve the change state identifier for the update. A response message is always sent.

**Result**

The following is an example of a response message:

```xml
<BusinessObject1ByIDResponse_sync xmlns:nm="http://sap.com/xi/SAPGlobal20/Global">
  <BusinessObject1>
    <SAP_UUID>00163e01-27e0-1ed0-a6f9-129eeb25f05e</SAP_UUID>
    <id>1</id>
    <item>Washing Powder</item>
  </BusinessObject1>
  <Log />
</BusinessObject1ByIDResponse_sync>
```

Note that response and confirmation messages may contain an XML namespace definition for xmlns:prx="urn:sap.com:proxy...". This is only used for support purposes and should ignored at runtime.

### Create Operation

The create operation creates a business object instance. A confirmation message is always sent.

**Result**

The following is an example of a confirmation message:
Update Operation

The update operation updates a single instance of an object with checks for concurrent updates. The part of the business object susceptible to the update can be limited by the defined business object view. The update operation may add and/or remove parts of the business object but it is not meant to be used for status changes. A confirmation message is always sent.

The update operation checks for concurrent updates. It checks the `ChangeStateId` element to determine whether somebody else has changed the same business object instance since the last read. If this is the case, then an error message is returned.

Result

The following is an example of a response message:

```xml
<BusinessObject1viewCreateConfirmation_sync xmlns:nm="http://sap.com/xi/SAPGlobal20/Global">
  <BusinessObject1>
    <SAP_UUID>00163e01-27e0-1ed0-a6f9-a08d04afd1f8</SAP_UUID>
    <id>3</id>
  </BusinessObject1>
  <Log>
    <MaximumLogItemSeverityCode>1</MaximumLogItemSeverityCode>
    <Item>
      <TypeId>/CSG_RUNTIME/</TypeId>
      <SeverityCode>1</SeverityCode>
      <Note>Create operation was successful</Note>
    </Item>
  </Log>
</BusinessObject1viewCreateConfirmation_sync>
```

Query Operation

When defining a query operation in the Web Service Creation Wizard, you can only use queries that already exist for the business object. The query operation retrieves specified information in business object instances. Query operations only read data; there is no change of any persistent data. A response message is always sent. The basic query naming pattern is:

`Find[View]SimpleBy[Selection criteria]`

Where:
• The business object view defines the response structure.
• The selection criteria view splits the operation so that each operation offers a semantically meaningful set of selection criteria typically needed for a given purpose. It is determined from the name of the underlying business object query.

**Maximum Number of Rows Returned**

You can define the maximum number of rows returned by a query using the following elements of the `ProcessingConditions` element:

- `QueryHitsUnlimitedIndicator`
- `QueryHitsMaximumNumberValue`

If the `QueryHitsUnlimitedIndicator` is not set and the `QueryHitsMaximumNumberValue` is zero, the default of 100 rows is assumed.

⚠️ The caller of the Web service is responsible for creating a reasonable package size. Therefore, we do not recommend that you set the unlimited indicator to true or that you return more than 1000 rows.

Note that the `LastReturnedObjectID` element is not supported in query response messages. Queries return a filtered result and may offer sorting criteria. Therefore, choosing a key or ID will depend on the query parameters, the returned fields, and the sorting order used in the query results. Also, not all business object nodes will have a key or the key will not have a linear order, for example if it is a UUID. The caller has to decide which fields can be used in each specific case. In simple cases the ID element of the node for which the query is defined is sufficient. In complex cases, all elements of the response have to be treated as part of the key.

**Comparisons Permitted**

You can use the following types of comparisons in Web service query operations:

- Equals
  
  If an Equals string contains an asterisk (*), the system implicitly interprets this as a wildcard query and thus as a CONTAINS pattern. You cannot therefore query items that have the asterisk in their name. For example, `QUERY PRODUCT WHERE PRODUCT ID=A*` returns all products beginning with the letter A. It does not return the details of the product name A*.

- Between
- Less than
- Less than or equal to
- Greater than
- Greater than or equal to

Each query parameter includes a `SelectionBy[Name]`, where the name determines the query parameter, and has the following structure:

**Details of Query Parameters**

<table>
<thead>
<tr>
<th>Element</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InclusionExclusionCode</td>
<td>I</td>
<td>Include in the result set</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>Exclude from the result set</td>
</tr>
<tr>
<td>Element</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IntervalBoundaryTypeCode</td>
<td>1</td>
<td>Equal to Single Value; = X</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Between (excluding upper boundary) Interval with closed lower and open upper boundary; ([X, Y))</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Between Interval with open lower and open upper boundary; ((X, Y))</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Between (excluding both boundaries) Interval with open lower and open upper boundary; ((X, Y))</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Between (excluding lower boundary) Interval with open lower and closed upper boundary; ((X, Y])</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Less than Interval with unlimited lower and open upper boundary; &lt; X</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Less Than or Equal to Interval with unlimited lower and closed upper boundary; (&lt;= X)</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Greater Than Interval with open lower and unlimited upper boundary; &gt; X</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Greater Than or Equal to Interval with closed lower and unlimited upper boundary; (&gt;= X)</td>
</tr>
</tbody>
</table>

**Result**

The following is an example of a response message:

```xml
<nm:BusinessObject1QueryByElementsSimpleByConfirmation_sync xmlns:nm="http://sap.com/xi/SAPGlobal20/Global">
  <BusinessObject1>
    <SAP_UUID>00163e01-3318-1ed0-b4df-43192913f03f</SAP_UUID>
    <ID>1</ID>
    <FirstName>Abc</FirstName>
    <MiddleName>D</MiddleName>
    <LastName>Xyz</LastName>
    <State>1</State>
  </BusinessObject1>
  <ProcessingConditions>
    <ReturnedQueryHitsNumberValue>1</ReturnedQueryHitsNumberValue>
    <MoreHitsAvailableIndicator>false</MoreHitsAvailableIndicator>
  </ProcessingConditions>
  <Log>
    <MaximumLogItemSeverityCode>1</MaximumLogItemSeverityCode>
    <Item>
      <TypeID>009</TypeID>
      <SeverityCode>1</SeverityCode>
      <Note>Query returned 1 records</Note>
    </Item>
  </Log>
</nm:BusinessObject1QueryByElementsSimpleByConfirmation_sync>```
**Action Operation**

When defining an action operation in the *Web Service Creation Wizard*, you can only use actions that already exist for the business object. The action operation changes the state of a business object instance. It is not intended to be used to modify data. The request message types are typically very short; they often contain just the business object ID. A response message is always sent.

**Result**

The following example shows an action response message:

```xml
  <Log>
    <MaximumLogItemSeverityCode>1</MaximumLogItemSeverityCode>
    <Item>
      <TypeId>017(/CSG_RUNTIME/)</TypeId>
      <SeverityCode>1</SeverityCode>
      <Note>Action RESETSTATE992EBFB4F6ADFACD executed</Note>
    </Item>
  </Log>
</nm:BusinessObject1ResetStateResetStateConfirmation_sync>
```

**See Also**

*Web Services Quick Guide* [page 480]

6.12.3 Tasks

6.12.3.1 Create a Web Service

**Overview**

You can create a Web service based on an SAP business object or one of your business objects. You select the business object elements and define the service operations that carry out tasks on the business object, for example, creating or updating the business object data.

The Web service can then be consumed by a client application to access the business object data remotely.

**Prerequisites**

- If you create a Web service based on an SAP business object, the business object must be released for the SAP public solution model (PSM).
- If you create a Web service based on your business object, the business object must be activated.
Procedure

1. In the Solution Explorer, do one of the following:
   - If you create a Web service based on an SAP business object or any of your business objects, proceed as follows:
     a. Right-click the project in your solution and select Add New Item.
     b. Select the Web Service template, enter a name and click Add.
   - If you create a Web service based on one of your business objects, right-click the business object and select Create Web Service.

   The Web Service Creation Wizard opens.

2. In the Enter Web Service Name step, proceed as follows:
   - If you used the Add New Item function, select a namespace and the business object based on which you want to create the Web service.
   - If the Web service is based on your business object, enter a name for the Web service.

3. In the Select Elements step, select the business object elements that you need for the Web service operations.

   If you have already created a Web service for the business object, you must enter a business object view name.

   If you create a Web service based on a business object that has an association to another business object, you can also add elements of the associated business object. For more information, see Association [page 160].

   If you have included a dependent object in your business object definition, you can only select the dependent object node and not the elements. The elements of the dependent object, however, are included. For more information on dependent objects, see here [page 163].

4. Define the service operations for the Web service. For each service operation that you define, select the checkbox, enter a name, and select on which business object elements you want to perform the service operation.

   The following Web service operations are provided by the SDK:
   - Create service operation
   - Read service operation
   - Update service operation
   - Query service operation
   - Action service operation

   For more information about service operations, see Web Service Operations [page 487].

5. In the Define Web Service Authorization step, assign your service operations to work center views. This will authorize business users who are assigned to the views to use the service operations in the Web service.

   Proceed as follows:
   a. Click Add and enter a name for an existing work center view or a new view. If you enter a name for a new work center view, the view is created when you have completed all steps in the wizard.
   b. For each work center view, select the service operations that you want to assign.
For more information about the authorization concept, see Web Services [page 483] under Authorization Concept.

6. Review your settings and click Finish.
A Web service with the file extension .webservice appears in the Solution Explorer. Below the Web service, the Web service authorization object with the extension .wsauth is listed, which contains the authorization settings of the Web service.

7. In the Solution Explorer, right-click the Web service and select Activate.
You need to adjust the derived names in the WSDL file if you are prompted to do so. You can also optionally choose to adjust the names for ease of use. For more information, see Web Services Quick Guide [page 480] under Adjust Derived Names.

Result
The Web service and the Web service authorization object are activated. You can now download the WSDL file, which can be consumed by a client application.

If you entered a name for a new work center view while defining the Web service authorization settings, the view appears in the Solution Explorer with the file extension .uiwocview. To edit the work center view, right-click the file and select Open in UI Designer. For more information, see Model a Work Center View [page 523].

See Also
Test a Web Service [page 496]

6.12.3.2 Test a Web Service

Overview
Before you can use your Web service in a client application, you must test it in a separate testing tool.

Prerequisites
- You have created and activated the Web service and have defined the Web service authorizations. For more information, see Create a Web Service [page 494].
- To download the WSDL file, the Web service must be in scope.
If you have created BAC elements in your solution and assigned your solution content to a business option, you can perform scoping manually in the SAP Business ByDesign development environment. You must be assigned the Business User role and have access rights for the Business Configuration work center. For more information, see User Setup Quick Guide [page 39]. Alternatively, you can trigger the deployment of business configuration in the Solution Explorer by right-clicking the Business Configuration node and selecting Deploy Business Configuration.
When you use the Deploy Business Configuration function, the system assigns all solution content that you have created to a default business option. To simplify testing of the solution, this default business option is always activated. However, you cannot test partial activation of business configuration content in the development environment because the Deploy Business Configuration function deploys all business configuration content independently of any business options you have created.

Procedure

1. In the SDK, in the Solution Explorer, right-click the Web service that you want to test and select Download WSDL File.

   If an error occurs when you download the WSDL file, check that your user has been created according to the procedure described in the User Setup Quick Guide [page 39].

2. Import the WSDL file into your client testing tool.

3. In the SAP solution, assign a test user to the work center views authorized for the Web service.

   Note that the work center views that are authorized to access the Web service must have been assigned to a work center. You can then assign test users to this work center in the system for the SAP solution. To do so, navigate to the User and Access Management work center, Business Users view. For more information, see the Business Users Quick Guide.

   This step needs to be repeated when your customers go live with your final solution. You have to ensure that any users who should have access to the Web service are assigned to the relevant work center views in the customer's productive system.

4. Test the Web service.

   For more information about the syntax used, see Web Service Operations [page 487].

See Also

Modify a Web Service [page 497]
Web Services [page 483]

6.12.3.3 Modify a Web Service

Overview

You can modify a Web service that you have created based on an SAP business object or one of your business objects. To modify a Web service, you can do the following:

- Add or remove business object elements for the service operations already defined in the Web service.
- Add or remove service operations.
- Adjust the derived names for the message data types used in the WSDL file.
Prerequisites
You have created a Web service based on an SAP business object or one of your business objects. For more information, see Create a Web Service [page 494].

Procedure
1. In the Solution Explorer, double-click the .webservice file of the Web service that you want to modify.
2. To modify your Web service, do one of the following:
   - To add or remove a business object element available for use in the Web service, select the Business Object View tab. Select or deselect the element that you want to add or remove. If you deselect a business object element, it will be removed from all existing service operations. If you add an element, it will become available for use in all existing service operations.
   - To add or remove a business object element from a specific service operation, select the Service Operations tab. Proceed as follows:
     a. From the dropdown list, select the service operation from which you want to add or remove a business object element.
     b. Select or deselect the relevant business object element.
   - To add or delete a service operation, select the Service Operations tab.
     a. Click Add.
     b. Select a service operation and enter a unique name for the service operation. Click Add.
     - To delete a service operation from your Web service, proceed as follows:
       a. From the dropdown list, select the service operation that you want to delete.
       b. Click Delete.
   - To adjust the derived name for a message data type used in the WSDL file, select the Message Data Types tab. When you create a Web service, the corresponding request and response messages for each service operation contain derived names based on the names of the business object elements and Web service operations. You may be prompted to adjust these derived names if they exceed the 120 characters limit. You can also optionally choose to adjust the names for ease of use. To adjust the derived name, proceed as follows:
     a. Select a service operation and then choose either the request or response message type. The derived names used in the message are displayed.
     b. Select a line in the structure and enter an alternative name in the Abbreviated Name field.
     c. Click Apply.
     - The new name is displayed in the message structure. If you renamed a line in the structure that has subitems, then the subitems are renamed accordingly.
     - If you want to reset a derived name of a message type to its initial name, select the service operation and choose the request or response message type. Click Reset Tree to Initial Names.
3. Click Save.
4. In the Solution Explorer, right-click the Web service and select Activate.

When you re-activate an existing Web service in the SDK, an updated version of the Web service's WSDL file is generated. You must download the new WSDL file after you re-activate the Web service.

Result

The modified Web service is activated. The system updates the Web service authorization object with the changes you made to existing service operations. However, if you have added new service operations, you must edit the authorizations manually. For more information, see Web Services Quick Guide [page 480] under Edit Web Service Authorizations.

You can now download the updated WSDL file, which then can be consumed by a client application to access the business object data remotely.

6.12.3.4 Integrate an External Web Service

Overview

You can integrate an external Web service into your solution by importing the WSDL file of this Web service. You can then address the Web service in your business logic to enable your solution to access a remote application using SOAP-based communication.

Prerequisites

- You have opened a solution in the SDK.
- You have saved the WSDL file of the Web service that you want to integrate into your solution on your local computer.
- If the WSDL file refers to one or more XML Schema documents, you have saved the schema documents on your local computer.
- If the WSDL file refers to XML Schema documents and the value of the schemaLocation attribute in the WSDL file is a URL, you have replaced the URL with the location of the XML Schema document on your computer. Note that the document’s location must be relative to the location of the WSDL file.
For more information about WSDL and XML Schema documents, see Web Services [page 483].

Procedure

1. Right-click the project in your solution and select Add New Item.
2. Select the External Web Service Integration template, enter a name and click Add.
   The External Web Service Integration Wizard opens.
3. In the Import WSDL File step, process as follows:
a. Select the WSDL file of the Web service that you want to integrate into your solution.
b. Select the Use Session ID checkbox, if the external web service allows stateful communication, that is, returns a session ID/cookie when it is called.

4. In the Select XML Schema Document step, proceed as follows:
a. Optional: If the WSDL file imported in the previous step refers to one or more XML Schema documents, select the XML schema documents required checkbox. Select the XML Schema document from your local computer and click Add Document.
b. Click Next.

5. Review your settings and click Finish.

An external Web service integration file with the extension .wsid appears in the Solution Explorer. Below the Web service, the WSDL file of the external Web service with the extension .wsdl is listed.

6. In the Solution Explorer, right-click the external Web service integration and select Activate.

Result

The external Web service integration is activated. You can add it as an outbound service in a communication scenario and create a communication arrangement based on the communication scenario. For more information, see Define a Communication Scenario [page 503].

For information on how to integrate an external http Web service into your solution, see Allow Nonsecure URLs (HTTP) [page 500].

See Also

Web Services Quick Guide [page 480]

6.12.3.5 Allow Nonsecure URLs (HTTP)

You can allow the usage of external http web services in an SAP Business ByDesign system. When you define the external web service integration scenario in the SAP Business ByDesign studio, you select the Allow Usage of Nonsecure URLs (HTTP) option. In SAP Business ByDesign, users are then able to define a communication arrangement using the http protocol.

Prerequisites

- You have a user in the SAP Business ByDesign studio and in SAP Business ByDesign.
- In SAP Business ByDesign, the Application and User Management work center has been assigned to your user.
- You are familiar with integrating external web services into solutions. For more information, see Integrate an External Web Service [page 499].
- You have opened a solution in the SAP Business ByDesign studio.
Procedure

1. Create a communication scenario in the SAP Business ByDesign studio.
   a. In the Solution Explorer, click the Add New Item button.
   b. In the Add New Item dialog, select SAP Service Integration > Communication Scenario, enter a name and click Add. The Communication Scenario Definition Wizard opens.
   c. Enter a name for the communication scenario. Select Application Integration and Allow Nonsecure URLs (HTTP).
   d. On the following screen, select the outbound service and click Next and Finish.

2. Create a communication system in SAP Business ByDesign.
   b. In the Communication Systems view, select New and make the following entries:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>&lt;specify an ID&gt;</td>
</tr>
<tr>
<td>Host Name</td>
<td>&lt;specify the URL of the Web service&gt;</td>
</tr>
<tr>
<td>System Access Type</td>
<td>Internet</td>
</tr>
</tbody>
</table>

   c. Select Actions > Set to Active and then Save and Close.

3. Create a communication arrangement in SAP Business ByDesign.
   b. In the Communication Arrangements view, select New.
   c. Select the communication scenario you created and click Next.
   d. Select your system instance ID and click Next.
   e. Click Edit Advanced Settings and deselect the Use Basic Settings option.
   f. In the Application Protocol field, select http and in the Path field, select your web service.
   g. Select Next, Finish, and Close.

6.12.3.6 Modify an External Web Service Integration

Overview

You can modify an external Web service integration that you have created to access a remote application using SOAP-based communication. You can modify an external Web service integration as follows:

- Replace the existing WSDL file with a new file.
- Add or remove XML schema documents, if any are referenced in the WSDL file.
Prerequisites

- You have created an external Web service integration. For more information, see Integrate an External Web Service [page 499].
- If you want to replace the current WSDL file with a new file, you have saved the new WSDL file on your local computer.
- If the WSDL file refers to one or more XML Schema documents, you have saved the schema documents on your local computer.

Procedure

1. In the Solution Explorer, double-click the .wsid file of the external Web service integration that you want to modify.
2. To modify your external Web service integration, select the WSDL File Details tab. Do one of the following:
   - To import a new WSDL file, click Upload WSDL File and select the file from the location on your computer where you have saved it.
   - If the WSDL file refers to one or more XML Schema documents, select the schema document from your local computer and click Add.
   - If you want to remove one or more XML Schema documents that you have already added, select the schema document from the list and click Remove.
3. To view the service operations provided by the external Web service, select the Service Operations tab.
4. Optional: To test the execution of the service operations provided by the external Web service, select the Test Tool tab.
   For more information, see Web Services Quick Guide [page 480] under Test a Service Operation.
5. Click Save.
   If you import a new WSDL file and XML Schema documents were uploaded for the previous WSDL file, the schema documents are deleted on saving the changes.
6. In the Solution Explorer, right-click the external Web service integration file, and select Activate.

Result

The external Web service integration is re-activated. You can add it as an outbound service in a communication scenario and create a communication arrangement based on the communication scenario. For more information, see Define a Communication Scenario [page 503].

See Also

Web Services Quick Guide [page 480]
6.12.3.7 Define a Communication Scenario

Overview

In a communication scenario, you can define a set of inbound and outbound services for a business process. Based on a communication scenario, a customer can create a communication arrangement in the SAP solution to exchange business documents electronically.

For more information about communication arrangements, see Communication Arrangements Quick Guide.

Prerequisites

- You have opened a solution in the SDK.
- You have created an inbound service or an outbound service. An inbound service is a Web service that is based on a business object. An outbound service is a Web service that you have integrated into your solution. For more information about creating a Web service, see Create a Web Service [page 494]. For more information about integrating an external Web service, see Integrate an External Web Service [page 499].

When you define a communication scenario, you must add at least one inbound service or one outbound service.

Procedure

1. Right-click the project in your solution and select Add New Item.
2. Select the Communication Scenario template, enter a name, and click Add. The Communication Scenario Definition Wizard opens.
3. In the Enter Communication Scenario Name step, select the communication type and enter a name for the communication scenario. For more information about communication types, see Communication Arrangements Quick Guide under Communication Arrangements.
4. In the Select Services and Operations step, select the inbound and outbound services that you want to include in your communication scenario.
5. Review your settings and click Finish. A communication scenario definition with the file extension .csd appears in the Solution Explorer.
6. In the Solution Explorer, right-click the communication scenario definition and select Activate.
7. Log on to the SAP solution as a key user and create a communication arrangement. For more information, see Communication Arrangements Quick Guide under Create a Communication Arrangement.

You can also create a communication arrangement or a communication system in the SAP solution by doing one of the following:

- Right-click the activated communication scenario definition and select Manage Communication Arrangement. The Communication Arrangements view in the SAP solution opens.
If the communication scenario is based on an application integration communication type, right-click the activated communication scenario definition and select Manage Communication System. The Communication Systems view in the SAP solution opens. For more information about communication systems, see Communication Systems Quick Guide.

8. Optional: Test the execution of the service operations provided by the external Web service that you have integrated into your solution.
   For more information, see Web Services Quick Guide [page 480] under Test a Service Operation.

9. Call the service operations of the inbound and outbound services in a script file.
   For more information, see Define the Business Logic to Call a Service Operation [page 505].

See Also
Web Services Quick Guide [page 480]

6.12.3.8 Modify a Communication Scenario

Overview
You can modify a communication scenario that you have created based on a business-to-business or application integration communication type. You can modify a communication scenario as follows:

- Change the communication type on which the communication scenario is based.
- Add or remove inbound and outbound services.

Prerequisites
You have defined a communication scenario. For more information, see Define a Communication Scenario [page 503].

Procedure
1. In the Solution Explorer, double-click the .csd file of the communication scenario that you want to modify.
2. To modify your communication scenario, do one of the following:
   - Change the communication type on which the communication scenario is based.
   - Select or deselect the inbound and outbound services that you want to add or remove.
3. Click Save.
4. In the Solution Explorer, right-click the communication scenario and select Activate.

Result
The communication scenario definition is re-activated. You can now use the modified communication scenario to create a communication arrangement in the SAP solution.
For more information, see Communication Arrangements Quick Guide under Create a Communication Arrangement.
6.12.3.9 Define the Business Logic to Call a Service Operation

Overview

If you have integrated an external Web service into your solution, you typically call the service operations provided by the Web service in a script file. The external Web service integration generates a library in the project namespace. This library is similar to a reuse library and you can use it to implement your business logic to call the service operations.

Prerequisites

- You have included the external Web service integration as an outbound service in your communication scenario. For more information, see Define a Communication Scenario [page 503].
- You have created a communication arrangement in the SAP solution. For more information, see Communication Arrangements Quick Guide under Create a Communication Arrangement.

Procedure

1. Create a business object.
   
   The following is an example of a business object definition in a solution that consumes an external Web service, which contains a service operation `Sum` that calculates the sum of two numbers:

   ```java
   import AP.Common.GDT as apCommonGDT;

   businessobject CalculatorBO raises CommunicationFault
   {
       message CommunicationFault text "Logical Port Does not Exist";
       element firstNumber : Identifier;
       element secondNumber : Identifier;
       element resultSum : Identifier;
   }
   ```

   For more information, see Business Objects Quick Guide [page 297].

2. Create screens for the business object.
   
   For the example in the previous step, you can create an Object Work List (OWL) and a Quick Activity Floorplan (QAF). The QAF can accept two numbers and return their sum by calling the external Web service.

   For more information, see Generate Screens for a Business Object [page 314].

3. Define the business logic for the business object to call the service operations provided by the external Web service.
For the example in the first step, you can create a BeforeSave script file. For more information, see Sample Code: Event BeforeSave Web Service Operation [page 506].

**Result**

The external Web service has been addressed in your business logic. You can now log on to the SAP solution and test the solution to verify whether the external Web service integration was successful.

**See Also**

Web Services Quick Guide [page 480]

6.12.4 Sample Code: Event BeforeSave Web Service Operation

```absl
/*
   This ABSL script is being implementated for:
   - Business Object : CalculatorBO
   - Node : Root
   - Event : BeforeSave

   In this example:
   - The request message contains two variables: pintVar1 and pintVar2
   - The response message contains a variable SumResult that returns the sum of two numbers and a CommunicationFault message

*/

import ABSL;

// Prepare the request variable requestSum
var requestSum : Library::Calculator.Sum.Request;

// Assign the business object elements to the request variables pintVar1 and pintVar2
requestSum.pintVar1 = this.firstNumber;
requestSum.pintVar2 = this.secondNumber;

// Call the external web service Calculator in the format (<request variable>,"<business partner ID>","<communication scenario name>") and prepare the response variable
var responseSum = Library::Calculator.Sum(requestSum, "MC1000", "CalculatorScenario");

// Capture the communication faults that occurred during execution
var commFault = response.CommunicationFault;

// Address the individual fault items and raise messages
var commFaultItem = commFault.Item;

foreach ( var item in commFaultItem )
{
    raise CommunicationFault.Create("E");
    return;
}
```
// Retrieve the sum from the response variable and assign its value to the business object element resultSum
this.resultSum = responseSum.SumResult;
7 User Interface Designer

7.1 Overview of the User Interface Designer

The user interface designer (UI designer) encompasses the following reference material:

<table>
<thead>
<tr>
<th>Get Informed About the UI Designer</th>
<th>Floorplan Modeling  [page 512] Get an overview of the steps involved in modeling different types of floorplans.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to the UI Designer</td>
<td>Floorplan Modification  [page 532] Get informed about the modifications and enhancements you can make to the floorplans you created.</td>
</tr>
<tr>
<td>Features of the UI Designer</td>
<td>Change Transactions  [page 562] Get an overview of the changes you can make to SAP floorplans. These changes are termed as change transactions and need anchors to be executed.</td>
</tr>
<tr>
<td>Application Programming Interface for SDK Custom Panes and Custom Controls  [page 433] Learn about the API that enables you to access Controller.DataContainer to read the data from the actual running client component and register to events in case of changes, as well as visualize data on your UI in the manner you choose</td>
<td>Mobile Floorplans  [page 574] Get an overview about how to adapt floorplans for use on mobile devices and the steps involved in configuring navigation between floorplans.</td>
</tr>
</tbody>
</table>

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

7.2 Getting Started with the User Interface Designer

7.2.1 Introduction to the User Interface Designer

Overview

The user interface designer (UID) is used to design the user interface (UI) of an application. You can use it to enhance and extend the user interface of the SDK application. The UID is integrated into the SDK, and you can use it to create new screens, or modify the screens provided by SAP. The UID can also be used to design screens for use on mobile devices.

You can create the following screens for a particular business object (BO) using the UID:

- Fact Sheet floorplan
- Quick Activity floorplan
Object Instance floorplan
Object Work list
Object Value selector

For more information, see Floorplan Modeling Quick Guide [page 512].

You also have the option to create screens with navigation. For more information, see Generate Screens for a Business Object [page 314].

You can modify the screens that you have created by adding controls and by creating event handlers, queries, interfaces and navigations. You can bind the controls you have added to BO elements so that when a particular control is used data is retrieved from the business object.

For more information, see Floorplan Modification Quick Guide [page 532].

The modifications that you can make to SAP floorplans are predefined by SAP and controlled by the presence of stable anchors. For more information, see Anchors [page 564] and Change Transactions Quick Guide [page 562].

You can also modify floorplans for use on a mobile device. For more information, see Mobile Floorplans Quick Guide [page 574].

Prerequisites

- You have created a solution in the SDK.
- All the required application entities such as the business object and the embedded component have been modeled in the solution.

Process Flow

1. Create user interface components such as an object work list, fact sheet, quick activity floorplan, or an object value selector dialog box.
2. Drag and drop the required user interface controls to design the layout of the screen.
3. Define the properties of each user interface control.
4. Bind the business object elements with the user interface elements.
5. Define the behavior of the application by creating event handlers, queries, interfaces and navigations.
6. Preview the application to check your changes.
7. Save and activate the component.

7.2.2 Features of the User Interface Designer

Overview

This document provides a brief overview of the common features available in the user interface designer (UID) to develop the user interface.

File Menu

<table>
<thead>
<tr>
<th>Menu Entry</th>
<th>Description</th>
</tr>
</thead>
</table>

SAP Business ByDesign Studio — FP4.0
User Interface Designer

PUBLIC • © 2012 SAP AG. All rights reserved. 509
<table>
<thead>
<tr>
<th>Close</th>
<th>Closes a user interface component.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import</td>
<td>Allows you to import user interface component from a file in the local disk. File can be of type .UserInterfacecomponent file or a .xml file.</td>
</tr>
<tr>
<td>Save</td>
<td>Saves a user interface component in the repository.</td>
</tr>
<tr>
<td>Save All</td>
<td>Saves all the user interface components that are being edited.</td>
</tr>
<tr>
<td>Activate</td>
<td>Activates a user interface component, so that the model is available to all users who are accessing the repository.</td>
</tr>
<tr>
<td>Exit</td>
<td>Exits the User Interface Designer.</td>
</tr>
</tbody>
</table>

**View Menu**

The View menu allows you to display or close the following tool windows:

- **Configuration Explorer**
  The Configuration Explorer displays the repository content in the UID. You can filter the content by a selected component type. The last level in the hierarchy is the user interface component. You can double-click to open the component. The tool tip for the component shows the component type and the complete path which acts as a unique identifier.

- **BO Browser/Data Model**
  The BO Browser appears in different places in the UID:
  - As part of the BOBrowser / Datamodel tool
  - In the Data Model tab of a component
  - In the Controller tab of a component

  In the BO Browser tab, the assigned BO or ECO is displayed as a tree structure with all available nodes and fields. You can:
  - Choose to display only the elements, queries, actions or everything
  - Add elements and attributes, create associations, queries, actions and their parameters
  - Reset the client side cached metadata from the backend (in case the ECO/BO has been updated)
  - Filter for specific categories (if allowed)
  - Search for opened nodes in the hierarchy

  In the Data Model tab you can:
  - Create the structure of the data model
  - Create, delete and rename nodes in the data model structure
  - Bind the data model element to the business object element
  - Display the corresponding properties of the nodes in the Properties Explorer

  In the DataModel tab the data structure is displayed together with various data model elements. The binding status of each of these elements is represented using specific colors. You can also determine the binding status for each element in the Properties Explorer.

<table>
<thead>
<tr>
<th>Color</th>
<th>Binding Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>Indicates an internal data model element. These are used internally for different purposes. Binding Status in the Properties Explorer has been set to Internal.</td>
</tr>
<tr>
<td>Yellow</td>
<td>Indicates an unbound data model element. Data binding has not yet been specified for this element. Binding Status in the Properties Explorer has been set to ToBind.</td>
</tr>
</tbody>
</table>
Green Indicates a bound data model element. Valid data binding has been specified for this element, or Binding Status in the Properties Explorer has been set to OK.

Red Indicates a data model element for which the data binding has to be corrected. Binding Status in the Properties Explorer has been set to ToCorrect.

- **Properties Explorer**
  It allows you to view and change properties for each element in the user interface model.
  As soon as you select an element in the Designer tab, the DataModel tab or the Controller tab, the corresponding properties of the selected element are displayed in the Properties Explorer. You can edit a few properties, and some are read only.

- **Toolbox**
  The Toolbox provides the panes and controls that can be used to design the layout of a user interface component in the Designer tab.
  The controls and panes available depend on the floorplan. You can drag the panes and controls from the toolbox to the Designer tab.

- **Document Outline**
  The Document Outline provides an overview of all the controls used in the user interface of the selected component.

- **Project Context**
  The Project Context provides an overview of the users involved in the creation and use of a particular floorplan.

- **Check Results**
  Check Results provides the list of errors in the floorplan.

- **Where Used List**
  The Where Used List gives an idea about where exactly a particular user interface component is being used.

- **Output Window**
  The Output Window gives an overview of all the changes made to the floorplan that is currently open.

- **Version History**
  The Version History gives an idea about the name and type of solution the user is working in, the date and time when the solution was last modified and the user who is responsible for the modifications.

- **OBN Explorer**
  The OBN Explorer will provide information about the operations, target floorplans and the inports configured in a particular BO.
  You can drag and drop the required OBN onto a user interface control in a source floorplan. The event handler and outport are automatically created; the outport will however need to be configured.

- **Extensibility Explorer**
  The Extensibility Explorer is used to modify floorplans that come from layer lower than the layer the user is working on. It displays the anchors assigned for a particular area, such the toolbar, section group, panecontainer etc and the change transactions associated with these anchors.
  All the change transactions that have been made can be viewed in the Extensibility Explorer. Change transactions once created, are displayed in the Unsaved Change Transaction section, once they are saved they are displayed in the Saved Change Transaction section.
  After activation, the change transaction are displayed in the Published Change Transaction section.

These tool windows can be moved around, docked or hidden.
Tools Menu

<table>
<thead>
<tr>
<th>Menu Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update Metadata</td>
<td>Updates the backend meta data information in the active component.</td>
</tr>
<tr>
<td>Options...</td>
<td>Allows the user to personalize the debug and warning message settings.</td>
</tr>
<tr>
<td>Cleanup UI Component</td>
<td>Allows the user to delete any unused data structures created by default along with Action Parameters.</td>
</tr>
<tr>
<td>Add to favorite</td>
<td>Adds the floorplan to a favorite list.</td>
</tr>
</tbody>
</table>

Content Area

The content area has the following tabs:

- **Designer**: To design the user interface of an application with the available controls from the Toolbox.
- **Data Model**: To bind the user interface elements with a business object element.
- **Controller**: To define the interaction between user interface components by creating event handlers, interfaces, navigations and queries.
- **Controller**: To preview the changes made to an user interface component, say a floorplan.

7.3 Modeling Floorplans

7.3.1 Floorplan Modeling Quick Guide

This document gives you an overview of the steps involved in modeling different types of floorplans in the SDK using the User Interface Designer.

In the SDK, you can generate screens for the business objects in your solution.

For more information, see *Generate Screens for a Business Object* [page 314].

Business and technical background

**SAP’s User Interface Style Guide**

You can learn how to build user interfaces according to this style guide.

The procedures are available in the Business Center at [https://www.sme.sap.com](https://www.sme.sap.com), under ▶️ *SAP Business ByDesign* ▶️ *Community* ▶️ *Wiki Info Exchange* ▶️ *SAP Solutions OnDemand Studio* in the section ▶️ *Partner/ Developer Information* ▶️ *Public Solution Model*.

Here you can also find a list of all current user experience rules. These are intended as a support for you to achieve a similar user experience with regard to interaction, look and feel as in the standard SAP solution. The rules are included in an automatic check tool inside the User Interface Designer.
Tasks

Model a Factsheet
You can create a fact sheet to display a read only overview of an object.
For more information, see here [page 515].

Model a Quick Activity floorplan
You can create a quick activity to display an overview of a business object.
For more information, see here [page 516].

Model an Object Instance floorplan
An object instance floorplan (OIF) is used to create and edit business object data or trigger actions on a business object. It is opened using object-based navigation. It can be used to create new business object data. OIFs are similar to quick activities except that an OIF has a set of tabs such as General and Line Items.
For more information, see here [page 517].

Model an Object Work List
You can create a list pane that is always embedded in a work center view. The OWL contains business objects such as purchase orders or opportunities. You can, for example, search for, display, and edit business objects as well as trigger follow-on processes.
For more information, see here [page 518].

Create an Object Value Selector Dialog Box
You can create an object value selector (OVS) dialog box as a reusable component. It can be called from any OVS control.
An OVS helps you to select one item from a table of business object instances.
For more information, see here [page 520].

Model a Guided Activity Floorplan
You can create a guided activity floorplan to simplify complex tasks. This floorplan helps you navigate step by step through a given process.
For more information, see here [page 521].

Model a Work Center View
You can model a Work Center View to hold a list of related floorplans. The Work Center View is a component that can be used across different Work Center floorplans that allow easy navigation to different sections from within the specified floorplan.
For more information, see here [page 523].

Model a Work Center
You can model a work center floorplan to display the various work center views.
For more information, see here [page 525].
Create a Port Type Package

You can create a port type package in the SDK and modify its attributes in the User Interface Designer. A port type package is a component that hosts one or more port types. A port type is a blueprint for an inport or an outport. Navigation between components that have inports and outports within the same port type package can be facilitated without any mapping effort.

For more information, see here [page 526].

Model a Modal Dialog

You can model a modal dialog to create a dialog box that is triggered by an event action. You can model a reusable modal dialog and then bind it to a floorplan, or you can model a modal dialog that is specific to a floorplan and that can be called by an event handler.

For more information, see here [page 527].

Model an Embedded Component

Embedded components are reusable subscreens that facilitate:

- Inport and outport navigation between the parent component and the embedded component.
- Bindings to reference data from the parent component.
- Configuration to let the parent component influence the behavior of an embedded component.

For more information, see here [page 529].

Create a Business Configuration View

You can create a business configuration view (BC view) for a business configuration set (BC set) to allow key users to add new values during fine-tuning or to delete the values of a BC set. You create the view in the studio and you model the view in the UI designer.

You can only create a BC view for a business configuration set (BC set) that was created using a custom business configuration object (BCO). You cannot create a BC view for an SAP BCO.

For more information, see here [page 295].

Save and Activate Your Changes

After you have created any of the floorplans listed above, you have to save and activate the floorplans.

1. From the File menu, choose Save.
2. From the File menu, choose Activate.
3. In the Activate Worklist dialog box, select the check boxes beside the relevant components and click OK.
7.3.2 Tasks

7.3.2.1 Model a Fact Sheet

Overview

You can create a fact sheet to display a read-only overview of an object. If you want to allow editing, you must create a quick activity floorplan or an object instance floorplan.

Procedure

1. In the Solution Explorer, expand the solution to display the business object.
2. Right-click on the business object and select Create Screens.
3. In the Create New Screen dialog box, do the following:
   a. In the Screen Name field, enter a name for the fact sheet.
   b. Under FloorplanType, select Fact Sheet (FS).
   c. Click OK. A new fact sheet is created and displayed in the Solution Explorer under the business object.
   By default, the static text controls are available in the identification region (below the title of the fact sheet).
   If required, you can change the component type from Fact Sheet (FS) to Quick Activity (QA) in the Properties Explorer, under Component Info-Component Type.
5. Click Display <-> Edit.
6. Select the identification region.
7. Enter a title for the fact sheet in the Properties Explorer, under Misc - FloorplanTitle.
8. Drag the required controls from the Toolbox to the fact sheet. For example, fields required for the general information, supplier, and project assignment of a contract.
9. Do the following for each of the required static text fields that are available below the fact sheet title:
   a. Select the static text field.
   b. In the Properties Explorer, under Data Information – Text, click the arrow button.
   c. In the Dependent Property Editor, select the required element and click OK.
   d. If you do not want to use the associated back end text for a static text field, enter a name for the field in the Properties Explorer, under Text Information – Label.
10. In the Data Model tab, all the elements of the selected business object are displayed. Bind each of the data model elements to a business object element. For more information, see Bind Data Elements to Business Object Elements [page 540].
11. Create and configure the required event handlers in the Controller tab. For example, create and configure event handlers for each of the buttons available on the fact sheet. For more information, see Create an Event Handler [page 554].
12. If you want to provide value help for any of the fields, see Configure an Object Value Selector [page 556].
13. To enable the fact sheet to be opened using object-based navigation, configure the fact sheet so that it can receive parameters from any floorplan.
   For more information, see Configure an Object-Based Navigation in a Target Floorplan [page 550].
14. Save and activate your changes.

Example
You want to create a fact sheet that shows the following information:
- Main details such as the contract status and the supplier
- General information such as contract validity period, person responsible and expiry status
- Supplier information
- Project assignment details

7.3.2.2 Model a Quick Activity Floorplan

Overview
You can create a quick activity to display an overview of a business object. You can further view all the details of the business object in the associated object instance floorplan.

Prerequisites
- You have created the required business object.

Procedure
1. In the Solution Explorer, expand the solution to display the business object.
2. Right-click on the business object and select Create Screens.
3. In the Create New Screen dialog box, do the following:
   a. In the Screen Name field, enter a name for the quick activity.
   b. Under FloorplanType select Quick Activity floorplan (QAF).
   c. Click OK.
      A new quick activity is created and displayed in the Solution Explorer under the business object.
4. Double-click the new QAF.
   The QAF opens in the User Interface Designer. It has the following attributes:
   - An identification region with the title of the quick activity
   - A contextual navigation region with default buttons such as Close, Save and Close, and View All
   - An event handler with the operation type WindowAction named Close associated with the Close button
5. Click Display <-> Edit.
6. Enter a title for the quick activity in the Properties Explorer, under Misc – FloorplanTitle.
   For example, if this quick activity allows you to:
7. Configure the View All button to navigate to the corresponding object instance floorplan to view the details of the object.
For more information, see Configure an Object-Based Navigation in a Source Floorplan [page 551].
9. Drag the required controls from the Toolbox to the quick activity.
10. In the Data Model tab, bind each of the data model elements to a business object element.
For more information, see Bind Data Elements to Business Object Elements [page 540].
11. Create an OBN inport to receive the navigation key from the source floorplan, in order to retrieve data from
the target floorplan:
   a. In the Controller tab, right-click on Imports and choose Add Import.
   b. Check OBN Inport.
   c. Select an operation, PortType Package and a PortType Reference as OpenByKey.
   d. Add the parameter Key List and Key and bind the parameter to an unbound data list and data field that acts as a placeholder for the key.
   e. Create an event handler as follows:
      a. In the Operation Type field, select BOOperation.
      b. In the BO Operation Type select Read.
      c. In the Navigation Type select Key Navigation.
      d. Bind the previously used, unbound data field in the ... column.
   f. In the Properties Explorer for the newly created inport, under Events — OnFire, select the even handler you have created.
12. Save and activate your changes.

7.3.2.3 Model an Object Instance Floorplan

Overview
An object instance floorplan (OIF) is used to create and edit business object data or trigger actions on a business object. It is opened using object-based navigation. It can be used to create new business object data. OIFs are similar to quick activities except that an OIF has a set of tabs such as General and Line Items.
Procedure

1. In the **Solution Explorer**, expand the solution to display the business object.
2. Right-click on the business object and select **Create Screens**.
3. In the **Create New Screen** dialog box, do the following:
   a. In the **Screen Name** field, enter a name for the object instance floorplan.
   b. Under **FloorplanType** select **Object Instance Floorplan (OIF)**.
   c. Click **OK**
      A new object instance floorplan is created and displayed in the **Solution Explorer** under the business object.
4. Double-click the new OIF.
   The OIF opens in the User Interface Designer.
5. Click **Display <-> Edit**.
6. In the **Designer** tab, select an OIF tab, for example, **General Data** tab.
7. In the **Properties Explorer**, select **ViewSwitchNavigation** from the drop-down list and under **Misc-NavigationItems**, click **...**.
8. In the **OIF View Collection Editor**, remove any unwanted tabs.
9. Edit the properties of each of the tabs. For example, the title of the tab, its behavior, and visibility.

   The name of the tab is stored in run time in the data model field /Root/UIState/ViewSwitchSelectedItem. You can use it in dynamic data or property handling for a calculation rule.

10. If you want to add sub tabs to a tab, do the following:
   a. In the **Properties Explorer**, select **ViewSwitchNavigationItem:NavigationItem** and under **Misc-ChildNavigationItems** click **...**.
   b. In the **OIF SubView Collection Editor**, click **Add**.
   c. Edit the properties of the sub tab, such as the title.
11. Design each of the tabs and sub tabs, and configure the OIF similar to a quick activity.
    For more information, see steps 7 to 11 of **Model a Quick Activity Floorplan** [page 516].
12. Save and activate your changes.

7.3.2.4 Model an Object Work List

Overview

You can create a list pane that is always embedded in a work center view. The OWL contains business objects such as purchase orders or opportunities. You can, for example, search for, display, and edit business objects as well as trigger follow-on processes.

Prerequisites

You have created an object instance floorplan (OIF), or a fact sheet.

For more information, see **Model an Object Instance Floorplan** [page 517] or **Model a Fact Sheet** [page 515].
Steps

1. Create an object work list (OWL)
   a. In the Solution Explorer, expand the solution to display the business object.
   b. Right-click on the business object and select Create Screens.
   c. In the Create New Screen dialog box, do the following:
      ● In the Screen Name field, enter a name for the object work list.
      ● Under FloorplanType select Object Work List (OWL).
      ● Click OK.
      A new object work list is created and displayed in the Solution Explorer under the business object.
   d. Double-click the new OWL.
      The OWL opens in the User Interface Designer.

2. Configure the data model view of the OWL
   a. Click Display <-> Edit.
   b. Add columns in the OWL by dragging and dropping Controls from the Toolbox.
   c. Bind each of the fields to the required business object element.
      For more information, see Bind Data Elements to Business Object Elements [page 540].

3. Configure the design view of the OWL
   On the Designer tab, you can see that the following fields are available by default:
   ● List or advanced list pane with columns
   ● Show drop-down list, basic Find field, Go button in the Search area
   a. In the Properties Explorer, change the properties of the required controls.
   b. In the Properties Explorer, enable the preview area by selecting AdvancedListPaneVariant.
      Under Appearance-UsePreviewPane, select True.
   c. In the BO Browser/Data Model, add the necessary fields from the DataList.
   d. To add the search parameter fields in the find form area, do the following:
      1. Click Advanced.
      2. In the BO Browser tool window set a filter to show queries.
      3. Drag query parameters to the find form.
      If you add the query parameters like this, then you do not need to add the query parameters and do the binding in the Controller tab.
   e. To configure a button to open target floor plan from the OWL, for example, an OIF or a GAF do the following:
      1. Add an Edit button to the tool bar
         For more information, see Configure a Button [page 541]
      2. Select the Navigation property and open the editor.
      3. Select the namespace, OIF/FS controller, and its import operation Open and click OK.
      4. An OBN, an outport, and an event handler are created automatically in the Controller tab.

4. Configure the controller tab of the OWL
a. In the **Controller** tab there is a default query. You can add more queries and default sets using the context menu.
b. In the **Designer** tab, assign a title to the default set that is displayed in the **Show** drop down list.
c. In the **Controller** tab, bind the query to the controller’s query using the elements structure.
d. Add and bind the query parameters from the controller to **Search Parameter Structure** fields.
e. Find the event handler, outport, and OBN that were created while configuring the **Edit** button.
f. To configure the outport, add the parameters **Key List** and **Key** and bind the parameter to an unbound data list and data field that acts as a placeholder for the key.
g. Save and activate your changes.

### 7.3.2.5 Create an Object Value Selector Dialog Box

#### Overview
You can create an object value selector (OVS) dialog box as a reusable component. It can be called from any OVS control. An OVS helps you to select one item from a table of business object instances.

#### Prerequisites
You have created the required business object.

#### Procedure
1. In the **Solution Explorer**, expand the solution to display the business object.
2. Right-click on the business object and select **Create Screens**.
3. In the **Create New Screen** dialog box, do the following:
   a. In the **Screen Name** field, enter a name for the OVS dialog box.
   b. Under **FloorplanType** select **Object Value Selector (OVS)**.
4. In the **Select Business Object Element for OVS** dialog box, select the business object element and click **OK**. A new OVS dialog box is created and displayed in the **Solution Explorer** under the business object.
5. Double-click the new OVS.
   The OVS opens in the User Interface Designer and has the following attributes:
   - Event handlers: Used to close the window, to retrieve the value set, to handle the selection change and to return the selected value
   - Query to display the queried business data
   - Inport: Used to access the OVS
   - Outport: Used to transfer the required data from the OVS to the floorplan where it is being used.
6. Click **Display <-> Edit**.
7. Select identification region in the in the OVS dialog.
8. In the **Properties Explorer**, under **Misc-Window Title**, click the arrow button.
9. In the **Dependent Property Editor**, enter the title and click **OK**.
10. In the **Data Model** tab, bind the following:
- Root to the corresponding OVS (enhanced) controller object
- List to the ECO node
- List attributes to the ECO node attributes

For more information, see Bind Data Elements to Business Object Elements [page 540].

11. In the Controller tab do the following:
- Select the event handlers GetValueHelpHandler and SetValueHelpHandler, and define the appropriate business object model.
- Select the query of the data model and bind it to the business object.
- Select the outport OVSValueSelectorOutPort to define the correct parameter binding from the parameter Object ID and Object Name to the attributes of the data model.

12. Save and activate your changes.

Follow-On Processes or Tasks

After you have created an OVS dialog box, you can create an OVS control. For more information, see Configure an OVS Control [page 556].

7.3.2.6 Modal a Guided Activity Floorplan

Overview

You can create a guided activity floorplan to simplify complex tasks. This floorplan helps you navigate step by step through a given process.

Procedure

1. Create a guided activity.
   a. In the Solution Explorer, right-click your project and select Add △ New Item.
   b. The Add New Item dialog box opens.
   c. Select Guided Activity Floorplan (GAF).
   d. In the Name field, enter a name for the guided activity floorplan.
   e. Click Add.
      The guided activity floorplan is added to your project.
   f. Double-click the guided activity floorplan.
      The guided activity floorplan opens in the User Interface Designer.
   g. Click Display<->Edit.
   h. A new guided activity shows the following:
      • An IdentificationRegion with the title of the floorplan
      • A RoadmapNavigation region with standard steps such as Step, Review and Confirmation
      • A ComponentToolbar with default buttons such as Previous, Next, Finish, and Cancel.

2. Design the layout and bind the data elements.
   a. Rename each of the steps:
1. On the **Designer** tab, select a step in the road map navigation region.
2. In the **Properties Explorer**, under **Design — Title**, click the arrow button. The **Dependent Property Editor** opens.
3. In the **Overridden Text** field, enter the name for the step.
4. Click **OK**.
5. Under **Misc — Name**, enter the name of the step.

The name of the current active tab is stored in the data model field `/Root/UIState/ViewSwitchSelectedItem`. You can use it in dynamic data or property handling for calculation rules.

b. If required add additional steps by right-clicking on the RoadmapNavigation region.

c. Drag the required controls from the **Toolbox** to each step in the guided activity.

d. In the **Data Model** tab, bind each of the data model elements to a business object element. For more information, see [Bind Data Elements to Business Object Elements](page 540).

3. **Configure the navigation buttons.**

   a. In the **Controller** tab, create two event handlers with operation type **Script**, say **HandlePrev** and **HandleNext**.

   b. Write a ruby script such that when the event handlers are called, they allow navigation to the next step.

   c. Under **Scope Fields** select a data field as an invalidation trigger, such that whenever the value of the field changes the script is executed.

   d. Assign the event handlers to the **OnClick** event of the **Previous** and **Next** buttons respectively.

   e. To dynamically enable and display the **Previous, Next, Finish** and other available buttons, do the following for each button:

      1. Select the button.
      2. In the **Properties Explorer**, under **Behavior — Enabled**, select **Advanced**.
      3. In the **Dependent Property Editor**, select **Calculation Rule**.
      4. Write a calculation rule to control the **Enabled** and **Visible** properties of the button.

4. **Configure the steps.**

   a. Configure the **Review** step:

      - Display an overview of the steps performed, with read only controls and transparent tables.
      - Disable the **Next** button.
      - Enable the **Finish** button and configure it such that on click it shows the **Confirmation** step.

   b. Configure the **Confirmation** step:

      - Hide the **Cancel** and **Finish** buttons.
      - Show the **Close** button.
      - Add a section group to the form region; with the header, **What do you want to do next?**
      - Add **Links** to the section group and configure the navigation.

        For example, the links could be to **Create another business object instance** or **Edit business object instance**.

   c. Configure the floorplan title, action buttons and create an OBN Import similar to a quick activity configuration.

For more information, see [Model a Quick Activity Floorplan](page 516).

5. **Save and activate your changes.**
7.3.2.7 Model a Work Center View

Prerequisites

- The required OWL has been created.
- If you want to create a list of common tasks, then the corresponding target floor plans, which implement the OBN operations used in this view, have to be declared as assigned objects.
- If there are common tasks and related links that cannot be specifically assigned to one view and belong to every view of a work center, an Include View with these links is created.

Procedure

1. Create a work center view.
   a. In the Solution Explorer, right-click your project and select Add New Item.
   b. The Add New Item dialog box opens.
   c. Select Workcenter View.
   d. In the Name field, enter a name for the work center view.
   e. Click Add.
      The work center view is added to your project.
   f. Double-click the work center view.
      The work center view opens in the User Interface Designer.
   g. Click Display<- Edit.
   h. Check the following properties of the work center view in the Properties Explorer:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration — Help ID</td>
<td>ID of the related help resource</td>
</tr>
<tr>
<td>Configuration — Support Area Code</td>
<td>Used in ITSAM, for example to propose a CSN component for an incident</td>
</tr>
<tr>
<td>Configuration — View SubType</td>
<td>Type of the view, for example, Overview, Application, IncludeView</td>
</tr>
<tr>
<td>Extensibility — Anchor</td>
<td>Associated flexibility anchors</td>
</tr>
<tr>
<td>Misc — Name</td>
<td>Name of the User Interface Component</td>
</tr>
<tr>
<td>RBAM Data — Authorization</td>
<td>Classifies the RBAM start authorization behavior. For normal views, the entry</td>
</tr>
<tr>
<td>Classification Code</td>
<td>should always be ApplicationContainer.</td>
</tr>
</tbody>
</table>

2. Embed an OWL.
   a. Drag and drop the OWL from the Configuration Explorer to the Designer tab.
   b. Click Bind.
   c. In the Embedded component binding dialog box, bind the OWL to the Workcenter View.
      For more information, see Model an Embedded Component [page 529].

3. Define the assigned objects.
   All the assigned objects together define the complete authorization granted to a user assigned to the view. Assigned object also specify the collection of valid navigation targets for a view and the elements that should appear in the Reports view of a work center.
a. In the Properties Explorer, select WorkCenterView.

b. Under RBAM Data — Assigned Objects, click ...

c. In the Add Assigned Objects dialog box, click Add and then click ... to select the required Component Id.

d. Select the required Assigned Object Type:
   - Navigation Target: To specify the UI components that should be reachable from this view.
   - WocViewInclude: To include views of type Include View.
   - TaskList View: A collection of BTM task types that should be supported by the view.
   - Reports view: To allow the view to show a list of reports in the work center.

   The reports are assigned in the Business Analytics work center.

e. If you want to define the dependency of the assigned objects, define the scope rules.

f. If there are a set of navigation targets that belong to many views of a work center, create an Include View and assign the Include View as an assigned object to this view. An Include View is built like any other view with a few exceptions like:
   1. You cannot include Embedded Components.
   2. The View sub-type has to be Include View.
   3. An Include View cannot have another Include View.
   4. Access context cannot be maintained for Include View.
   5. An assigned object of type Include View does not have scope rules.

g. Click OK.

4. Specify the common tasks and related links specific to this view.

a. In the Properties Explorer, select WorkCenterView.

b. Under Links — CommonTasks, click ....

c. In the NavigationListItem Collection Editor, click Add.

d. Under Data — Link, click ....

e. In the Model Navigation ... dialog box, select an existing OBN navigation or create a simple navigation. For example, you can select an existing OBN navigation for the common task Create a Sales Order.

f. You can also create the required event handlers for the common tasks. For more information, see Create an Event Handler [page 554].

g. If you want a list of related links in the work center view, repeat the above steps under Links — Related Links.

5. Define the enterprise search objects.

a. In the Properties Explorer, select WorkCenterView.

b. Under RBAM Data — Enterprise Search Objects, click ....

c. In the Enterprise Search Objects Editor, assign the required enterprise search controller objects to the view and click OK.

   The users assigned to this view are authorized to use Enterprise Search and the assigned objects are available in the drop down list of available enterprise searches.

6. Define the external applications.

a. Under RBAM Data — External Application Data, click ....

b. In the Select External Application Data dialog box, assign the external applications and click OK.

   The service interfaces in the assigned external applications can now be used by this view.

7. Specify the access context.
If the view supports instance restrictions, you can maintain access contexts.
   a. Under **RBAM Data — Access Context Code**, select the required access code of the view.

8. **Save and activate your changes.**

### 7.3.2.8 Model a Work Center

#### Overview

You can model a work center floorplan to display the various work center views.

#### Prerequisites

You have created the required work center views.

#### Procedure

1. **Create a work center with default views.**
   a. In the **Solution Explorer**, right-click your project and select **Add → New Item**.
   b. The **Add New Item** dialog box opens.
   c. Select **Work Center**.
   d. In the **Name** field, enter a name for the work center.
   e. Click **Add**.
      The work center is added to your project.
   f. Double-click the work center.
      The work center opens in the User Interface Designer.
   g. Click **Display<->Edit**.
   h. Check the following properties of the work center in the **Properties Explorer**:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration-Help ID</td>
<td>ID of the related Wektra help resource</td>
</tr>
<tr>
<td>Extensibility-Anchor</td>
<td>Associated flexibility anchors</td>
</tr>
<tr>
<td>Misc-Name</td>
<td>Name of the User Interface Component</td>
</tr>
<tr>
<td>RBAM Data-Authorization Classification Code</td>
<td>This indicates how the component is handled for startup authorization. If there is an embedded work center view the entry should always be <strong>Application Container</strong>.</td>
</tr>
</tbody>
</table>

2. **Assign work center views to the work center.**
   a. On the **Designer** tab, click on the folder icon.
   b. In the **Properties Explorer**, select **OberonCenterStructure**.
   c. Under **Misc-ViewSwitches**, click **...**.
   d. In the **View Switch Collection Editor**, add a new view switch for each of the work center views as follows:
      1. Click **Add**.
      2. Click on the work center views you have added and under **Misc- Name**, enter a name for the view switch.
      3. Click **OK**.
e. To add a sub view:
   1. In the View Switch Collection Editor, select the view switch.
   2. Under Misc-SubViewSwitches, click ...
   3. In the Sub ViewSwitch Collection Editor, click Add.
   4. Click on the sub views you have added and under Misc-Name, enter a name for the sub view switch.
   5. Repeat the steps for the required number of sub views.

f. To assign a work center view to a view switch:
   1. In the Designer tab, select the view switch.
   2. From the Configuration Explorer drag and drop the work center view, to the Designer tab.
   3. Repeat the steps for all the view switches and sub view switches.

3. Configure the work center.
   a. Specify the scope rules for the work center and the view switches:
      1. In the Properties Explorer, select the work center.
      2. Under Configuration-ScopeRules, click ...
      3. In the Scope Rules Editor, assign the required scope rules and click OK.
      4. Repeat the steps for all the view switches.

4. Save and activate your changes.

7.3.2.9 Create a Port Type Package

Overview
You can create a port type package in the SDK and modify its attributes in the User Interface Designer.
A port type package is a component that hosts one or more port types. A port type is a blueprint for an inport or an outport. Navigation between components that have inports and outports within the same port type package can be facilitated without any mapping effort.

Prerequisites
You have created a solution in the SDK.

Procedure
1. In the Solution Explorer, right-click your project and select Add New Item
2. The Add New Item dialog box opens.
3. Select Port Type Package.
4. In the Name field, enter a name for the port type package.
5. Click Add.
   The port type package is added to your project.
6. Double-click the port type package.
   The port type package opens in the User Interface Designer.
7. Click Display<>Edit.
8. Create the required structure, as follows:
   - To add a port type, select the root node and then click +.
   - To add a parameter, right click on the port type and choose Add Parameter.
   - To add a List Parameter, right click on the port type and choose Add List Parameter.
     You can add parameters under a list parameter.
   - To set a parameter as a key parameter, right click on the parameter and choose Set as Key.
9. To rename any entity in the port type package, right click on the entity and choose Rename. Type the required name and press Enter.
10. Save and activate your changes.

7.3.2.10 Model a Modal Dialog

Overview
You can model a modal dialog to create a dialog box that is triggered by an event action. You can model a reusable modal dialog and then bind it to a floorplan, or you can model a modal dialog that is specific to a floorplan and that can be called by an event handler.

Prerequisites
You have created a solution in the SDK.

A. Model a reusable modal dialog

Procedure
1. In the Solution Explorer, right-click your project and select Add New Item.
2. The Add New Item dialog box opens.
3. Select Modal Dialog Floorplan.
4. Click Add.
5. The modal dialog is added to your project.
6. Double-click the modal dialog.
7. The modal dialog opens in the User Interface Designer.
8. Click Display<->Edit.
9. Select the identification region.
10. In the Properties Explorer, under Misc – Window Title, click the arrow button.
11. In the Dependent Property Editor, enter the title and click OK.
12. On the Designer tab, drag the required controls from the Toolbox to the panes.
13. On the Data Model tab, bind each of the fields to the required business object element.
   For more information, see Bind Data Elements to Business Object Elements. [page 540]
14. Create the required event handlers.
   For more information, see Create an Event Handler [page 554].
15. Create the bindings in the modal dialog as follows:
   a. On the Controller tab, right-click on Bindings and add a binding.
   b. Select a field or a structure for the binding.
   c. Enter a Binding Name.
   d. Click Bind.

16. In the parent component, do the following:
   a. On the Controller tab, right click on Modal Dialog and add a modal dialog.
   b. In the Repository dialog box, navigate to the modal dialog that you have created.
   c. Click OK.
      The modal dialog is added to the parent component.

17. On the Controller tab of the parent component, click the new modal dialog and then under Binding Configuration do the following:
   a. In the Parent data model, select the field for the binding.
   b. In the Controller Interface for Embedded Component, select the binding.
   c. Click Bind.
      The modal dialog is now bound to the parent component. This facilitates the synchronous passage of data from the parent component to the modal dialog.

18. Save and activate your changes.

B. Model a modal dialog specific to a floorplan

Procedure

1. In the Solution Explorer, double-click the floorplan where you want to add a new modal dialog.
   The floorplan opens in the User Interface Designer.
2. Click Display<-> Edit.
3. From the Toolbox – Panes, drag and drop a ModalDialog onto the Designer tab.
4. In the new modal dialog, do the following:
   a. In the Properties Explorer, under Text Information – Title, click on the arrow button.
   b. In the Dependent Property Editor dialog box enter the required title in the Overridden text field.
   c. Click OK.
   d. On the Designer tab, drag the required controls from the Toolbox to the panes.
   e. On the Data Model tab, bind each of the fields to the required business object element.
      For more information, see Bind Data Elements to Business Object Elements [page 540].
5. On the Controller tab, right click on Event Handler and add a new event handler.
6. For that event handler do the following:
   1. Select operation type as Show Modal Dialog.
   2. Under Configure the Operation-Modal Dialog name select the modal dialog you have created.
7. Assign the event handler to a button or link.
   For more information, see Create an Event Handler [page 554].
8. Save and activate your changes.
7.3.2.11 Model An Embedded Component

Overview

Embedded components are reusable sub screens that facilitate:

- **Inport and outport navigation** between the parent component and the embedded component.
- **Bindings** to reference data from the parent component.
- **Configuration** to let the parent component influence the behavior of an embedded component.

An embedded component can be integrated with its parent component as follows:

- **Tight coupling**: This is handled by referencing a parent data model either explicitly by creating bindings in the embedded component interface, or implicitly by node reference binding. For example, tight coupling is used in custom panes, address maintenance, and business cards.

- **Loose coupling**: Embedded components are triggered in an inport context mapping, for example, by passing an ID. An embedded component can pass back parameters to a parent component using outport context mapping. For example, mashups and information snippets displayed in a factsheet. If an embedded component is loosely coupled, it should be self-contained and the data binding and controller logic needs to be defined in the embedded component.

Procedure

1. **Create an Embedded Component**
   a. In the Solution Explorer, right-click your project and select Add → New Item.
   b. The Add New Item dialog box opens.
   c. Select Embedded Component.
   d. In the Name field, enter a name for the embedded component.
   e. Click Add.
      The embedded component is added to your project.
   f. Double-click the embedded component.
      The embedded component opens in the User Interface Designer.
   g. Click Display<->Edit.

   You can model an embedded component just like any other UI component. The main difference is in the controller interface binding, controller configuration and node reference binding.

2. **Prepare to tightly couple an embedded component with its parent component.**
   a. To tightly couple an embedded component by explicit binding, in the embedded component do the following:
      1. In the Controller tab, right-click on Bindings and choose Add Binding.
      2. Select the newly created binding.
      3. Select a field or a structure for the binding.
      4. Click Bind.
      5. In the Binding field, enter a name.
      6. Save and activate your changes.
b. To tightly couple an embedded component by node reference binding, in the embedded component do the following:

1. Drag the required fields to the Designer tab.
2. On the DataModel tab, select an embedded component node, for example, Root.
3. In the Properties Explorer under Node Reference Binding, set Virtual Binding to true.
4. On the Controller tab, right-click on Bindings and choose Add NodeReference Binding.
5. Select NodeReference and select the node on which the Virtual Binding property was set to true.
6. Click Bind.
7. Save and activate your changes.

3. **Bind the embedded component.**

a. Explicit binding

1. Open the parent component in edit mode.
2. In the Configuration Explorer, navigate to the embedded component you created.
3. Drag and drop the embedded component to the Designer tab.
4. Click Bind.
5. In the Embedded component binding dialog, in the Bindings Configuration tab, select an attribute or structure from the Parent Data Model and select a binding under the Controller Interface of Embedded Component.

b. Node reference binding

1. Open the parent component in edit mode.
2. In the parent component, create a data structure.
3. In the Configuration Explorer, navigate to the embedded component you created.
4. Drag and drop the embedded component to the Designer tab.
5. Click Bind.
6. Bind the data structure created in the parent component to the node reference binding created in the embedded component.

4. **Configure the embedded component.**

An embedded component can expose configurations, to let the parent component influence the behavior of the embedded component, through configuration parameters and configuration list parameters.

a. Add the required configuration parameters:

1. In the Controller tab, under Configuration, right-click Parameters select Add ConfigParam
2. Choose either Value or Value Help.

   Data is passed from the parent component to the embedded component. Value is the default value to be used by the embedded component in case it is not specified by the parent component. Value Help is used when the options to be selected by the parent component are known and fixed. A default value can be selected in Value Help too.

   For example, if the configuration parameter is LayoutType, then the options such as TableLayout, NewsPaper Layout could be added as value help and the table layout could be set as default. In the Parent component you can change it to NewsPaper Layout.

   You can group a few configuration parameters in a configuration parameter structure.

3. Create a configuration parameter list under Configuration, by right-clicking on Parameter List. The MinOccurs field property has a fixed value of 1 in the case of a Configuration Parameter List.
4. Specify the minimum and maximum amount of entries allowed in the list, by setting the `minOccursField` and `maxOccursField` properties of the configuration parameter list. A data model entity is created for each configuration parameter, structure, and parameter list. You can configure their properties in the `Properties Explorer`.

5. **Configure the parent component.**
   a. To use the configuration parameters of the embedded component, open the parent component.
   b. Select the embedded component.
   c. In the `Properties Explorer`, under `Parameters — StructureParameters`, click ...
   d. In the `StructureParameter Collection Editor`, to add values for structure parameters do the following:
      1. Click `Add`.
      2. Select the added structure parameter and under `Misc-Name` enter the name.
      3. Under `Parameters-Parameter` click the arrow button.
      4. In the `ECParameterValueEditor` enter or choose the parameter value and click `OK`.
      5. In the `StructureParameter Collection Editor` click `OK`.
   e. Save and activate your changes.

**Alternatives**

You can also loosely couple an embedded component to a parent component.

1. Create an embedded component.
2. In the `Controller` tab, model the inports, in the followings manner:
   a. Right-click on `Inports` and add a new inport.
   b. Add a new parameter and bind this to a data field in the data structure.
3. Model the outports to pass back values in a similar manner as the inports.
4. Open the parent component.
5. In the `Controller` tab, model the outports, to access the embedded component.
6. Model the inports to pass back values.
7. In the `Configuration Explorer`, drag the embedded component to the `Designer` tab and click `Bind`.
8. In the `Embedded component binding` dialog, in the `Simple Navigation` tab, right-click on `Navigation` and select `Add Navigation`.
9. Map the parent component outport with the embedded component inport and bind the parameters.
12. Map the embedded component outport with the parent component inport and bind the parameters.
13. Click `OK`.
14. Save and activate your changes.

**Result**

You should note that once an embedded component is used in a parent component, any interface changes may be incompatible.
7.4 Modify Your Floorplans

7.4.1 Floorplan Modification Quick Guide

This document gives you an idea about the various modifications and enhancements you can make to the floorplans created by you using the User Interface Designer.

For more information about creating floorplans, see the Floorplan Modeling Quick Guide [page 512].

Business and technical background

SAP’s User Interface Style Guide

You can learn how to build user interfaces according to this style guide.


Here you can also find a list of all current user experience rules. These are intended as a support for you to achieve a similar user experience with regard to interaction, look and feel as in the standard SAP solution. The rules are included in an automatic check tool inside the User Interface Designer.

Transformations

A business object (BO) is modeled generically, and not for a specific user. The BO structure and node attributes, therefore do not always fit the needs of a specific user interface.

A transformation allows the conversion of a BO node attribute into a form that can be used on the user interface.

For more information, see here [page 535].

Tasks

Fields

Bind Data Elements to Business Object Elements

You can bind the data elements you have added to your floorplans, with the business object elements in the backend, so that data can be derived from the backend when a particular data element is used at runtime.

For more information, see here [page 540].

Define Dynamic Behavior of a Field

A dynamic expression is used to define the behavior of a field within a specific context. You can for example, define the visibility of a field or a group of fields depending on the content of another field on the screen.

For more information, see here [page 540].

Buttons
Configure a Button
You can add any required button on the application toolbar. The Cancel and Close buttons are configured in the system. If you add these buttons, the system automatically associates the event handler, to close the floorplan. If you want to any other button, you need to write the required event handler. You can also configure a button to open a floorplan by configuring an object based navigation between floorplans.
For more information, see here [page 541].

Create a Preview Button
You can create a Preview button and link it to a print form. Clicking the Preview button allows users to preview the form as a PDF and provides options to either save or print the form.
For more information, see here [page 542].

Screen Sections
Create a Pane in a Floorplan
You can create a pane to display different types of controls on a screen.
For more information, see here [page 545].

Create a Preview Pane
You can create a preview pane to preview information about a business object listed in a table.
For more information, see here [page 546].

Screen Layout and User Interface Texts
Change the Title of a Component
You can change the text of any component visible on the user interface.
For more information, see here [page 547].

Hide a User Interface Component
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. Select the user interface component you want to hide, for example, a section group, or a field.
4. In the Properties Explorer, under Behavior — Visible, set the value to False.
5. Save and activate your changes.

Create a Text in the Text Pool
For more information, see here [page 548].

Object Worklists (OWLs)
Configure a Query
You can create and configure a query to view specific instances of a business object in a floorplan.
For more information, see here [page 549].

Tables
Reorder Columns in a List or a Table

1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. In the Properties Explorer, under Child Elements – List Columns, click ...
4. In the Column Collection Editor, select the column you want to move.
5. Use the up and down arrows to move the column to the required position.
6. Save and activate your changes.

Navigation between floorplans

Configure an Object-Based Navigation in a Target Floorplan

You can use object-based navigation (OBN) to navigate between floorplans and pass parameters between floorplans. For more information, see here [page 550].

Configure an Object-Based Navigation in a Source Floorplan

You can use object-based navigation (OBN) to navigate between floorplans and pass parameters between floorplans. For more information, see here [page 551].

External Navigation

Add a Link to trigger an External URL

You can add a link that opens an external URL in your floorplans. For more information, see here [page 553].

Transformations

Configure a Dedicated Field

You can add a dedicated field to a data model and use a field transformation to calculate the value of this dedicated field from existing business object (BO) node attributes. For more information, see here [page 557].

Configure a Default Set Parameter using Transformations

You can use a transformation to calculate the value of a default set parameter. For more information, see here [page 557].

Additional Tasks

Write a Script

You can write front-end scripts while creating an event handler and while defining a calculation rule. For more information, see here [page 240].

Create an Event Handler

You can create an event handler to respond to user actions on the screen. For more information, see here [page 554].
Configure an Action Form

You can create action forms to use the action parameters of a business object (BO) node or an enhanced controller object (ECO) node. Action forms are always bound to a BO/ECO node’s ESI action that has parameters. The binding is done within the BO Action operation of an event handler. Action forms belong to a user interface component and cannot be reused in other user interface components.

For more information, see here [page 555].

Configure an Object Value Selector

You can configure an Object Value Selector (OVS) control to open an OVS dialog box.

For more information, see here [page 556].

Add a Report List View to a Work Center

You need to assign standard report list views, for example the List and Gallery standard reports list views, to a work center in the User Interface Designer.

For more information, see here [page 559].

Define an Anchor for a Floorplan

You can define anchors for your floorplans to allow end users to make changes to the floorplans provided by you.

For more information, see here [page 560].

Save and Activate Your Changes

After you have created any of the changes listed above, you have to save and activate the changes.

1. From the File menu, choose Save.
2. From the File menu, choose Activate.
3. In the Activate Worklist dialog box, select the check boxes beside the relevant components and click OK.

7.4.2 Business and technical background

7.4.2.1 Transformations

Overview

A business object (BO) is modeled generically, and not for a specific user. The BO structure and node attributes, therefore do not always fit the needs of a specific user interface.

For example, in the Purchasing contracts BO, a specific value needs to be entered to ensure the unlimited validity of contracts. So a date like 31/12/9999 is used for this purpose. On the user interface however, a checkbox to indicate unlimited validity is needed and not a set date.

A transformation allows the conversion of a BO node attribute into a form that can be used on the user interface.

You can use transformation rules to calculate the value of fields or parameters in the user interface data model that do not have direct counterparts within the BO node structure.

Examples of transformation rules include:
A transformation has one or many inbound parameters. The input for a transformation can either be a constant value, a binding against a field of a BO node structure, or an already existing data field from the user interface. The resulting outbound value is assigned to a target field or a parameter inside the user interface data model.

The type of transformation rule used depends on whether the value being generated is for use in a dedicated field or in a default set parameter.

For each namespace, a predefined set of transformation rules are available.

See Also

Configure a Dedicated Field  [page 557]
Configure a Default Set Parameter using Transformations  [page 557]

7.4.2.2 Operation Types

Overview

Operation types in the User Interface Designer allow you to assign an activity to an event handler. When creating an event handler  [page 554], you can select from one or more of the following operation types:

**BOAction** can be used to invoke a special function exposed by an ESR action. BOAction parameters can be bound to data fields in the data model to pass values to parameters. To enter values for the action parameters, you can configure an action form in the BOAction. Add the controls to the action form and bind the action parameters to the data fields bound to controls. On executing the event handler, the action form is displayed. Enter the action parameter values and click OK to execute the BOAction.

You can configure the following additional parameters for the BOAction:

- Instance Binding is the binding of an action against the node instance(s). It is required for list-binding.
- Save Before Execution indicates whether or not a save before the execution of the action is required, for example for one-click actions.
- Save After Execution indicates whether or not an immediate save is required.
- Continue on Reject allows the execution to continue even when the BOAction fails. If this flag is not checked, the event handler is stopped.
- Special Reference Action supports copy and paste for lists. The list operation CutCopy stores the selected instances in the buffer. To use this selection for the next reference action, pasting in this case, you can model a special reference BOAction by checking this flag. Then, the selected instances in the first operation, CutCopy in this case, are passed as parameters for this second reference action.

**BOOperation** is an implicit ESI-Action. This operation is supported by all business objects even though it is not explicitly modeled in ESR as an action. It is used to perform Read, Create, DelayedCreate and ValueHelp.

You can configure the following additional parameters for the BOOperation:

- Set Suppress Messages to true to suppress all messages invoked by triggering this operation in the backend. Otherwise, all ESI and backend messages will be displayed.
Select Namespace and Select BO describe the BO on which the operation is performed.

BO Operation parameters are defined by the user and not derived from ESR metadata. For example, for a read operation, you need to pass the ID value to indicate with which ID you want to invoke the read.

There are the following parameter types:
- The keyNavigation parameter is passed with special metadata, here the key flag and GDT metadata.
- With nodeID, the value of the node id is passed.
- The alternativeKey is modeled for some BOs as number ranges, for example. In the backend, they need special treatment before a read/retrieve can be done.

The value of the alternative key node is converted to node ID. The type of alternative key is indicated by the data field bound to the path attribute. This value is used at runtime to identify the type.

CheckAndDetermine offers a generic function for checking BOs, without the need for an explicit, modeled BO action. A check carried out with this operation triggers the validation of a whole BO instance, or a specific node with subnode tree. Error or warning messages are sent to the client if there are any inconsistencies.

This operation is generally used for Guided Activity floorplans, but may also be used in other applications where you want to implement a check button.

Condition is used as a branch control to call different operations based on a data model value or constant. You can specify a default option, as well as configure multiple select options with different compare types and values; and nested condition operations.

ContainerExit is an action defined on the exitClass and should only be used in special situations, namely, wherever a data structure has an exit class assigned.

You can configure any one of the actions in the action field, or create new ones manually. If you use an existing action, the system automatically sets the exit class property of the structure you have selected as path.

DataOperation allows you to modify the value of data model fields.
- Assign the value of a text pool entry to the target data field with assignText.
- Assign the value of the source data field to the target data field with assign.
- Assign a constant value to the target data field with assignConstant.
- Clear the value of the target data field with clear.

DocumentOutput configures Microsoft Excel and Microsoft Word downloads. The group ID is used by the Excel/Word export to identify the assigned template group.

ExecuteDefaultSet executes an ESI query. Set Blocking to block access to the UI during query execution.

FireEventHandler allows you to invoke other event handlers.

FireOutport is used to fire and pass data to the outport. Set TakeOverMessagesToNewWindow to carry forward messages from the previous save to a new window in Save and New scenarios.

List triggers operations related to the data list such as Add Row and Remove Row. Set Blocking to block access to the UI during the operation.

MessageBox configures decision pop-ups, for example Are you sure you want to delete? A deletion template is available, as well as other parameters to customize your configuration. Use List Binding to enable plural text and list count for multi-selection.

OpenLink opens a link in an external browser. The link value can be bound to a data field and URL.

Set the browser target features as follows:
- Default = empty
- Status = yes
- Toolbar = yes
- Menubar = yes
Resizable = yes

Set Process Protocol to explicitly switch off the protocol handling for the OpenLink operation. In this way, you ensure that file:// or https:// remains in the URL.

Script allows you to define front-end script for dynamic expressions. Set Blocking to block access to the UI during script execution.

In the Scope Fields section, you can configure data fields that trigger script execution. When a field value changes, the script is executed.

ShowModalDialog allows you to display an in-component modal dialog as pop-up. To configure it, drag and drop Modal Dialog from the toolbox to the User Interface Designer or create one from the selection proposal in this operation.

SynchDataContainer forces a backend round trip and flushes the data container at runtime. You can use this operation instead of, for example, provoking a round trip by flagging a dummy field with requiresRoundTrip and setting a value in this field.

When the Finalize flag is set, DoPostProcessing is invoked which results in the evaluation and sending of messages.

ValueHelp should be configured if an object value selector (OVS) component is involved.

- Set the Use Query flag to trigger the query configured as query attribute.
- Query Name is the query in the OVS to be called.
- For Operation Type, Get is for retrieving DefaultSets, and Set for passing to the parent component.
- Use Suppress Messages to suppress messages coming in the value help round trip.
- Parent Component Id and Parent Path are the parent component node information for the RetrieveValueSet call.
- Select Namespace and Select BO Model describe the OVS controller name.

In WindowAction, set TakeOverMessageInCallingWindow to Yes to show messages of the current operation in the calling window.

For Success Notification Port, configure the outport to be fired on save. The outport and inport at the source should be configured with the same local event name. In this way, when a quick activity is opened from a work list, then saved and closed, the system informs the port that the save is successful and the work list is refreshed.

Set Continue on Reject to continue with the execution even if the window action fails.

Controller

RestartClient clears details from the DataModel and reloads the screen from the backend. In this way, if you open a screen, make a change, then cancel the operation, the screen reverts to its state before the edit operation was performed. There is no configuration for this operation.

QCAfterSaveAction is configured in the Quick Create (QC) component after a WindowAction operation.

7.4.2.3 Message Mapping

Overview

A message is the text shown at the bottom of the user interface (UI) during runtime. It presents the user with a piece of information, an instruction, or a warning in a given situation. In SAP’s on-demand solutions messages are sent to the UI from the business object (BO) that the UI is based on. The messages are part of the infrastructure delivered by SAP or can be part of a custom BO.

In some cases, the original messages of the BO that are shown during runtime do not fit the context of the UI. For example, they have the wrong terminology or the terms used are not specific enough. In those cases, you can exchange messages or change the behavior of messages on the UI by using the message mapping offered by the UI.
designer. If the original message contains parameters you can also display a different message for each individual parameter value.

You map messages only within the context of a specified UI and not across the system. This means, if a message is shown in the context of two different UI models, you can map it for one UI model and this mapping does not affect the message in any other UI model.

Prerequisites

Message mapping in the UI designer is only available for BOs that contain messages.

Structure

In the UI designer, message mapping is located on the Controller tab of those UI components that directly communicate with the BO or the backend. On the Controller tab, the following sections deal with the messages of a BO:

- **Messages**
  In this section, the UI designer displays all messages associated with the BO that the currently selected UI component is based on.

- **Message Mappings**
  In this section, you can see the original messages associated with the BO and you can do the actual mapping. For more information, see Map a Message [page 560].

- **Message Mapping Overview**
  In this section, you see the original messages together with the mapped messages.

In the UI designer, you can use the following rules to deal with messages:

- **Replace**
  The original message is replaced by a new message. During runtime, the system displays the new message on the UI instead of the original message.

- **Ignore**
  The system ignores the original message and does not show it during runtime on the UI.

- **Original**
  The system displays the original message during runtime.

- **Aggregate**
  Any number of messages can be combined into one new message. The system displays this new message on the UI during runtime instead of the original messages.

See Also

Map a Message  [page 560]
7.4.3 Tasks

7.4.3.1 Bind Data Elements to Business Object Elements

Overview
You can bind the data elements you have added to your floorplans, with the business object (BO) elements in the backend, so that data can be derived from the backend when a particular data element is used at runtime.

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. In the Data Model tab, under the Root structure, add the necessary data fields, data structures, and data lists.
4. Bind the data structures and the data fields under the Root structure to the BO structure nodes and the BO elements.
   To bind a data element, select the required data element from the left list and select the required business object element from the right list and click Bind.
5. In the dialog box that opens, click Yes if you want the system to automatically use the text maintained for the BO element as the field label.
6. If you want to manually enter a field label in the User Interface Designer, in the Designer tab do the following:
   a. Double-click the user interface control.
      The Dependent Property Editor dialog box opens.
   b. Enter the field label in the Overridden Text field.
7. Save and activate your changes.

7.4.3.2 Define Dynamic Behavior of a Field

Overview
A dynamic expression is used to define the behavior of a field within a specific context. You can for example, define the visibility of a field or a group of fields depending on the content of another field on the screen.

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. Select the required field.
4. In the Properties Explorer, under Behavior – Visible, select Advanced.
5. In the Dependent Property Editor, select the variant Calculation Rule.

6. Under Calculation Rule, define the formula as a Boolean expression.
   For example, if the formula is the following:
   ```java
   ($Data.DiscountType == 2 ) .
   return true
   else
   return false
   ```
   Then, this field is displayed only when the Discount Type field has the value 2.

   To define the formula, you can use every field that is defined in the DataModel. The fields available in the DataModel are listed under the collection pattern Invalidation Trigger.

   For more information, see Write a Script [page 553].

7. Under Invalidation Trigger, mark each of the fields you have used within the formula and add it using the arrow buttons.

8. Click OK.

9. Save and activate your changes.

Example

There are many discount types available in the system. If the user selects the Discount Type Christmas, you want the Minimum Bill Amount field to be visible on the screen. This field is otherwise not visible if the user selects any other discount type. In such a case, you can define a dynamic expression for the Minimum Bill Amount field.

7.4.3.3 Configure a Button

Overview

You can add any required button on the application toolbar. The Cancel and Close buttons are configured in the system. If you add these buttons, the system automatically associates the event handler, to close the floorplan. If you want to any other button, you need to write the required event handler. You can also configure a button to open a floorplan by configuring an object based navigation between floorplans.

Prerequisites

The floorplan where you want the button is created.

Procedure

1. In the SDK, double-click the required floorplan in the Solution Explorer.
   The floorplan opens in the User Interface Designer.

2. Click Display <-> Edit.

3. Open the Configure Toolbar, as follows:
   a. In the Properties Explorer, select the AdvancedListPaneVariant in which you want to add a button.
b. Under Toolbar Information - Use Toolbar, select True.
c. Under Toolbar Information - ListPaneToolbar, click ...

4. Add the required button in the Configure Toolbar dialog, as follows:

<table>
<thead>
<tr>
<th>Task</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>To add a configured button</td>
<td>Under Available list, select Close or Cancel button, and click the right arrow button. The button is now available in the Current list.</td>
</tr>
</tbody>
</table>
| To add an application-specific button and configure it manually | 1. Under Available list, select the required button or the Application Specific button, and then click the right arrow button.  
2. Under Events - OnClick, select New Event Handler.  
For more information, see Create an Event Handler [page 554]. |
| To add a button and configure an object based navigation | 1. Under Available list, select the required button or the Application Specific button, and then click the right arrow button.  
2. Configure OBN navigation in the source and target floorplans.  
For more information, see Configure an Object-Based Navigation in a Source Floorplan [page 551] and Configure an Object-Based Navigation in a Target Floorplan [page 550]. |

5. Click OK.

6. Save and activate your changes.

7.4.3.4 Create a Preview Button

Overview
You can create a Preview button and link it to a print form. Clicking the Preview button allows users to preview the form as a PDF file and provides options to either save or print the form.

Prerequisites
- An activated floorplan exists for the business object.  
  For more information, see Model an Object Instance Floorplan [page 517], steps 1–3.
- An activated print form is associated with the business object.  
  For more information, see Create a Print Form [page 413].

Procedure
1. Open the floorplan in the user interface designer (UI designer) and create an outport.
   a. In the SDK, double-click the floorplan in the Solution Explorer.  
The floorplan opens in the UI designer.
   b. Click Display <-> Edit.
   c. On the DataModel tab, right-click on Root and select Add Data Field.
   d. In the Properties Explorer, set the following properties of the data field:  
      • Initial Value: True
- **Type:** Boolean
- **Name:** bool_true

e. On the **Controller** tab, right-click on **Outports** and then select **Add Outport**.
f. To rename the outport, right-click the outport and select **Rename**. Enter a name, for example, **PreviewOutport**.

2. **Add the outport parameters.**
   a. In the **Outport Configuration: OutPort** view, click the **Add Parameter** button.
   b. In the **ParameterName** field, enter **GroupName**.
   c. In the **Constant** field, enter the group code.

   You can find the group code in the SDK. In the Solution Explorer, double-click the .ftgd file. The group code is displayed as shown in the figure below.

   ![Group code displayed in the SDK](image)

   Group code displayed in the SDK

d. Click the **Add Parameter** button again.
e. In the **ParameterName** field, enter **ForceToFrontendPreview**.
f. Next to **ParameterBinding**, click **...**. The **Parameter Binding** dialog box opens.
g. Select the data field **bool_true** and **OK**, then save your changes.

   The output parameters have been entered as shown in the figure below.

   ![Outport parameters entered in the Parameters Binding dialog box](image)

   Outport parameters entered in the Parameters Binding dialog box

3. **Bind the outport and import parameters.**
   a. On the **Controller** tab, right-click on **Modal Dialogs** and then select **Add ModalDialog**.
   b. In the **Repository Browser** dialog box, navigate to the modal dialog named **Preview** by selecting **SAP_BYD_APPLICATION_UI → Reuse → OutputManagement → Preview** and click **OK**.
   c. Under **Modal Dialogs**, click **Preview**.

   The **Bindings Configuration/Navigation** dialog box opens as shown in the following figure.
Bind each of the Source Outport parameters with the corresponding Target Inport parameter and save your changes.

4. **Create an event handler to fire the outport.**
   a. On the **Controller** tab, right-click **Event Handlers** and select **Add EventHandler**. The new event handler is displayed under **Event Handlers**.
   b. Right-click the new event handler and select **Rename**; then enter a new name for the event handler, for example, **PreviewHandler**.
   c. Under **EventHandler: Operations**, click the **Add Operation** button.
   d. Under **Type**, click in the field and then select **FireOutport**.
   e. Under **Configure the operation**, select the outport you have created (for example, **PreviewOutPort**) as **OutPort** and save your changes.

5. **To ensure that the user can use the preview button, associate the new event handler with the OnClick event.**
   a. On the **Designer** tab, select the navigation region (see the following figure).

   ![Navigation region](image)

   b. In the **Properties Explorer**, click in the **Toolbar** field and click `[... `next to **Configure**. The **Configure Toolbar** dialog box opens.
   c. Under **Available, StandardButtons**, select **Preview**.
   d. Add the button to **Current** by clicking the arrow pointing to the right.
   e. On the right, under **Events**, select **OnClick** and then the event handler you have created (for example, **PreviewHandler**) and click **OK**. The **Preview** button has been added to the navigation region.
   f. Save your changes.
6. Test the Preview button in the SAP solution.
   a. Select the Preview tab next to the Controller tab.
   b. Log on to the SAP solution.
      The SAP solution opens and the Preview button you have added is displayed as shown in the following figure.

The Preview button in SAP Business ByDesign

c. Click Preview.
   The Preview window opens.

See Also
Print Forms Quick Guide [page 411]

7.4.3.5 Create a Pane in a Floorplan

Overview
You can create a pane to display different types of controls on a screen.

Prerequisites
You have created the floorplan to which you want to add the pane.

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer.
   The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. Select the pane, for example, the pane above which you want to add new fields.
   The pane is highlighted with a red border.
4. You have the following options:
   ● To insert or delete a row or a column, right-click on the pane, select Layout and then choose the appropriate option.
   ● To expand the pane over two columns, click the arrow button on the right side of the pane.
   ● To move the pane to the next row, select the pane and drag the pane to the required row.
5. Drag and drop the required controls from the Toolbox into the new pane.
6. Bind each of the fields to the required business object element.
   For more information, see Bind Data Elements to Business Object Elements [page 540].

7. Save and activate your changes.

### 7.4.3.6 Create a Preview Pane

**Overview**

You can create a preview pane to preview information about a business object listed in a table.

**Procedure**

1. In the SDK, double-click the required floorplan in the Solution Explorer.
   The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. On the Designer tab, select the advanced list pane.
4. In the Properties Explorer, under Misc – List Pane Variants, click the arrow button.
   The List Pane Variant Collection Editor opens.
5. Under Appearance, set Use Preview Pane to True.
   The list pane selected, is now expanded with two rows. The system also allows you to add controls in those rows.
6. Drag the required controls from the Toolbox Controls to the preview panes.
   When you drag and drop a control, a Section Group is automatically created. You can add multiple controls to a section group and give a name to the section group. For example, the name of the section group can be Invoice Address.
7. In the Properties Explorer, under Text Information – Field Group Header, enter the required section group name.
8. Bind each of the fields to the required business object element.
   For more information, see Bind Data Elements to Business Object Elements [page 540].
9. Save and activate your changes.

**Example**

You have created an Object Work List (OWL) which shows a list of customers. You now want to create a preview pane, to view the following invoice address details of a selected customer, without leaving the OWL:

- Number
- Street
- City
- Pin Code
- Phone Number

You also want to view the number of orders that have not been shipped or delivered yet.

For each customer, you can configure a preview pane and add various fields to display the above information.
7.4.3.7 Change the Title of a Component

Overview
You can change the text of any component visible on the user interface. For example, you can change the following:

- Field label or tab text
- Group Header: Heading given to a group of fields.
- Roll Over Explanation Text: For field names that are not easy to understand, you can provide an explanation text. The system underlines fields with explanation texts in green and the explanation is visible on mouse over.
- Tooltip: In cases where a field name is abbreviated or is an icon, you can provide a tooltip for the field.
- Explanation Text: A sentence or two that is displayed on the user interface with details such as how to perform a task.
- Title of a floorplan, for example, a work center view.

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. Double-click the text that you want to change. The Dependent Property Editor dialog box opens.
   
   You can also change the text in the Properties Explorer. For example:
   
   - You can change the explanation text visible on the user interface under Text Information – Explanation Area Text.
   - You can provide a rollover explanation text under Text Information – Explanation.

4. Enter the required text in the Overidden text field.

   All text options that can be changed for a particular component are displayed at the bottom of the Dependent Property Editor dialog box.

5. Click OK.
6. Save and activate your changes.

7.4.3.8 Hide a User Interface Component

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan is opened in the User Interface Designer.
2. Click Display <-> Edit.
3. Select the UI component you want to hide, for example, a section group, or a field.
4. In the Properties Explorer, under Behavior — Visible, set the value to False.
5. Save and activate your changes.

See Also
Change the Title of a Component [page 547]

7.4.3.9 Create a Text in the Text Pool

Overview
You can create text pools in order to collect all texts that you want to use for a particular user interface component, such as a floorplan.

The system-generated and the user-specific text are visible in the TextPool option in the Controller tab. The system-generated text is created by the tool. It is associated with a particular component, is marked in red and cannot be edited in the textpool.

The user-specific texts can be entered by the user and are mostly used in scripts. They are marked in black. You can reuse these texts when you are selecting a label for a control using the Reuse Text option. Based on the context, the texts with the correct text category are displayed on the user interface.

Prerequisites
You have created the floorplan for which you want to create the text pool entry.

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. On the Controller tab, click TextPool.
4. Select the required language.
5. To add a new text to the text pool:
   a. Click Add Text.
      The UUID is automatically generated.
   b. Select a Text Category based on where you would like to reuse this text.
      X indicates short text that is less than 120 characters and Y indicates long text.
6. To enter the reusable text click ...
   The Text Block Entry Editor dialog opens.
7. In the Text Block Entry Editor, you can do the following:
   • Edit the static text: Change the text and click OK.
   • Reuse the static text: Click on Reuse Text and select one of the texts shown in the list and click OK.
   • Add an advanced concatenation of static and dynamic text parts. Dynamic parts inside the text are bound to data fields and are calculated during runtime.
8. Save and activate your changes.

7.4.3.10 Configure a Query

Overview

You can create and configure a query to view specific instances of a business object in a floorplan. For example, in the Contracts OWL, the Show drop down list has entries such as Contracts in Preparation, and Released Contracts. These are the default sets under a query and are used to display a filtered set of contracts. You can create a query and default sets under that query. These default sets are then added to the drop down list.

Procedure

1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. Add a data field to create a reference to your data type, as follows:
   a. On the Data Model tab, right-click on root and then add a data structure.
   b. Right-click on the structure and then add a data field.
4. Configure the query, as follows:
   a. On the Controller tab, right-click on Queries and then choose Add Query.
   b. In the Query Configuration, select the required Namespace and BO.
   c. The queries modeled under the BO are displayed in a hierarchical manner. Expanding the query node in the tree will show all the query parameters.
   d. Select the required query and click Bind.
   e. Add the query parameters and select the data field that you created as the required value.
5. Configure the default set, as follows:
   a. Right-click on the query and then choose Add Default Set.
   b. To rename the default set, right-click on the default set and then choose Rename.
   c. In the Default Set Configuration, enter a Title for the default set. This value is displayed in the Show field.
   d. Select the query parameter you have added and define how you want the parameter to be used, by choosing the required Select Options and values. You can now use the default set in the OWL.
6. Select a default query, as follows:
   a. In the Properties Explorer, select the Advanced List Pane.
   b. Select the required default set, under Misc — Default DefaultSet.
   c. Under Misc — Default Set Mapping, click ...
   d. In the Default Set Mapping dialog box, move the default sets to the Binding area and click OK.
7. Save and activate your changes.

7.4.3.11 Reorder Columns in a List or a Table

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan is opened in the User Interface Designer.
2. Click Display <-> Edit.
3. Select the list.
4. In the Properties Explorer, under Child Elements – List Columns, click ...
5. In the Column Collection Editor, select the column you want to move.
6. Use the up and down arrows to move the column to the required position.
7. Save and activate your changes.

7.4.3.12 Configure an Object-Based Navigation in a Target Floorplan

Overview
You can use object-based navigation (OBN) to navigate between floorplans and pass parameters between floorplans. The following instructions are to be performed in the floorplan that you would like to display on the click of a button. For example, the target floorplan can be the Product floorplan that is displayed when you click the Product ID link.

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. On the Data Model tab, add a data field.
4. If required, rename the data field.
5. On the Controller tab, right click Inports and select Add Inport.
6. Check the OBN Inport check box.
7. Select the required namespace and business object (BO) model.
8. Under the BO, select the node for which the inport is being configured. If no node is selected, by default the root node will be chosen by the system.
9. Create an operation.
10. Select a PortTypepackage and PortTypeReference. The combination of business object, business object node, and the operation identifies the navigation target.
11. You can select one of the following port type references:
   - OpenByKey: Used at instance when you want to open the target floorplan for more information about an object in a read only mode. For example, the target floorplan is a customer fact sheet to view information about a customer.
• **CreateWithRef**: Used at instance when you want to create an object with reference to the source floorplan. For example, the target floorplan is a Purchase Order OIF that is called from a Contract floorplan, to create a purchase order with reference to a contract.

12. Bind the data field you just created as the parameter for **Key**.

13. Bind the parent of the data field as the parameter for **KeyList**.

14. On the **Controller** tab, right-click on **Event handlers** and select **Add Eventhandler**.

15. If required, rename the event handler.

16. Select the Operations - Type as **BOOperation**.

17. Define the following **BO Operations Parameters**:
   a. Enter a name.
   b. Select the **Type** as **keyNavigation**.
   c. Select the data field that you created as **Bind**.

18. Select the import you created.

19. In the **Properties Explorer**, under **Events – OnFire**, select the event handler you created.

20. Save and activate your changes.

**Example**

You are on the **Products** object work list (OWL) that displays a list of products, and you want to view the details of a selected product in the **Edit Product** OIF.

In the **Products** OWL, when a user does one of the following, you want to open the Edit Product OIF:
- Select a **Product ID**
- Selects a row and click **View or Edit**

To do this, you should configure an OBN in the source (OWL) and the target (OIF) floorplan.

7.4.3.13 Configure an Object-Based Navigation in a Source Floorplan

**Overview**

You can use object-based navigation (OBN) to navigate between floorplans and pass parameters between floorplans. Perform the following steps in the floorplan where you click a UI control to open the target floorplan.

**Procedure**

1. In the SDK, double-click the required floorplan in the **Solution Explorer**. The floorplan opens in the User Interface Designer.

2. Click **Display <-> Edit**.

3. Right-click the user interface control that should open the target floorplan and choose **Add OBN using Wizard**.

4. The **Configure OBN** dialog box opens.

5. Under **Select BO details**, do the following:
   a. You can enter the name of the OBN being configured or the default name will be used.
b. Select the namespace.
c. Select the BO model.
d. Select the node under the BO for which you want to configure the OBN. If no node is selected, the root node is chosen by default.
e. Click Next.

6. Under **Select Operation**, do the following:
   a. From the list given, select the operation which needs to be executed. Each operation will have an associated *PortTypePackage* and *PortTypeReference* which get configured when you choose that particular operation. The combination of business object, business object node and the operation identifies the navigational target.
   b. Click Next.

7. Under **Configure Outport Data**, do the following:
   a. You can enter the name for the *Outport* being configured or the default name will be used.
   b. Add the *Context Attribute* details.
      Navigation can be made specific for a particular target floorplan by defining the context attributes, distinguishing it from a floorplan from the same business object, business object node and having the same operation configured for it.
   c. Define the parameters you want to pass on to the target floorplan. The event handler is automatically created and bound to the outport.
      You can also rename the event handler which is being created.
   d. Click Finish.

8. To edit a previously configured OBN, right-click on the User Interface control for which the OBN was previously configured and choose **Edit OBN Using Wizard**.

9. Save and activate your changes.

**Alternatives**

1. In the SDK, double-click the required floorplan in the *Solution Explorer*.
   The floorplan is opened in the User Interface Designer.
2. Click *Display <-> Edit*.
3. Select the button that should open the target floorplan.
4. In the *Properties Explorer*, under *Menu Information – Navigation*, click ...
5. In the *Model Navigation* dialog box, do the following:
   a. Select the target business object.
   b. Select the BO node defined in the target floorplan.
   c. Select the *Operation* defined in the target floorplan.
   d. Click OK.
6. On the *Controller* tab, select the outport that was automatically created.
7. Define the parameter you want to hand over to the target floorplan.
   The event handler is automatically created and bound to the outport.
8. To open the target floorplan on click of a link in a table, do the following:
   a. Select a column.
   b. In the *Properties Explorer*, select the *Display Type* as Link.
   c. Under *Events – OnClick*, define an event handler that is configured with *FireOutport*, to open the target floorplan.
9. Save and activate your changes.

7.4.3.14 Add a Link to Trigger an External URL

**Overview**

You can add a link that opens an external URL in your floorplans.

**Procedure**

1. In the SDK, double-click the required floorplan in the **Solution Explorer**.
   The floorplan opens in the User Interface Designer.
2. Click **Display <-> Edit**.
3. Drag the **Link** control from the **Toolbox** to the **Designer** tab.
4. On the **Designer** tab, select the **Link** user interface control.
5. In the **Properties Explorer**, under **Events - onClick**, select **New Event Handler**.
6. In the **Configure Event Handler** dialog box, select the **Operation Type OpenLink**.
7. Enter the required URL in the **Uri** field.
8. Click **OK**.
9. Save and activate your changes.

7.4.3.15 Write a Script

**Overview**

You can write front-end scripts while creating an event handler and while defining a calculation rule.

**Procedure**

1. In the SDK, double-click the required floorplan in the **Solution Explorer**.
   The floorplan opens in the User Interface Designer.
2. Click **Display <-> Edit**.
3. In the **Designer** tab, select the user interface control to which you want to associate this script.
4. To write a script in an event handler, for example, to trigger an action on the click of a button, do the following:
   a. In the **Properties Explorer**, under **Events - onClick**, select **New Event Handler**.
   b. In the **Configure Event Handler** dialog, select the **Operation Type** as **Script**.
5. To write a script in a calculation rule, for example to control the visibility of a user interface control, do the following:
   a. In the **Properties Explorer**, under **Behavior — Visible**, select **Advanced**.
   b. In the **Dependent Property Editor**, select **Calculation Rule**.
6. Write the front-end script and click **OK**.
   In the script, you can:
7. Use the data elements defined in the Data Model tab.
8. Use the scope fields or the invalidation trigger fields to model the dependency between user interface controls.
9. Use the IntelliSense feature. For example you can select the data elements that are defined in the floorplan as you type the script.
10. Click ZoomOut to expand the editor for a bigger view.
11. Click Check Syntax to check if the script is syntactically correct.

For information, see Front-End Script Reference [page 240].

7. Save and activate your changes.

7.4.3.16 Create an Event Handler

Overview

You can create an event handler to respond to user actions on the screen. For example, if you want to close the window when the user clicks on the Close button, you can create an event handler for the Close button. You can do either of the following:

- Create the user interface control, for example, a button, and then create an event handler.
- Create the event handlers first and then associate the event handler to the required user interface control.

Procedure

1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. Select the control for which you want to add an event handler.
4. In the Properties Explorer, under Events – OnClick select New Event Handler.
5. Enter a name for the event handler.
6. Select the type of operation.
   - You can add more than one operation, for example, a ListOperation for an AddRow button, a BOAction for a Save button and a WindowAction for a Close button.
   - For more information, see Operation Types [page 536].
7. Click OK.
8. Save and activate your changes.

Alternatives

You can also create an event handler first and then associate it with the required user interface control.

1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan is opened in the User Interface Designer.
2. Click Display <-> Edit.
3. On the Controller tab, under the Tree view, right-click on EventHandlers and then choose Add EventHandler.
4. Enter a name for the event handler.
5. Select the type of operation.
6. On the Designer tab, select the UI control with which you want to associate the event handler.
7. In the Properties Explorer, under Events – OnClick, select the event handler you have created.
8. Save and activate your changes.

7.4.3.17 Configure an Action Form

Overview
You can create action forms to use the action parameters of a business object (BO) node or an enhanced controller object (ECO) node. Action forms are always bound to a BO/ECO node’s ESI action that has parameters. The binding is done within the BO Action operation of an event handler. Action forms belong to a user interface component and cannot be reused in other user interface components.

You can either create an action form and then configure an event handler for it, or you can create an action form for an existing event handler.

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. In the Toolbox, under Panes, drag ActionForm to the Designer tab.
4. Double-click the action form.
5. Set the following properties:
   - Events – onCancelActionForm: Create an event handler to handle the Cancel action in the action form. For more information, see Create An Event Handler [page 554].
   - Misc – Title: Enter a title for the action form.
   - Misc – PreventDisplay: It is usually set to False. You can display the action form, even if all the required parameters are not available at run time. In such a case, set it to True.
6. Drag the controls from the Toolbox to the action form.
7. Bind the controls to the fields which are bound to the action parameters.
8. Save and activate your changes.

Alternatives
You can also create an action form while configuring the event handler, which calls the corresponding ESI action.
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan is opened in the User Interface Designer.
2. Click Display <-> Edit.
3. In the Controller tab, select the event handler that triggers the ESI action.

If there is no operation that calls an ESI action, create a new operation of type BO Action and map the operation to the corresponding ECO/BO action.
4. If the action parameters are not yet bound to data model attributes, bind these attributes in the **Selected Action Parameters** section.

5. Under **Action Form**, select the new action form. The action form with all the ESI action parameters will be created.

6. In the **Designer** tab, double-click on the action form.

7. If you want to display additional information, add the required fields.

8. Save and activate your changes.

### 7.4.3.18 Configure an Object Value Selector

**Overview**

You can configure an Object Value Selector (OVS) control to open an OVS dialog box.

**Prerequisites**

You have created the required OVS dialog box.

For more information, see, *Create an Object Value Selector Dialog Box* [page 520].

**Procedure**

1. In the SDK, double-click the required floorplan in the **Solution Explorer**. The floorplan opens in the User Interface Designer.

2. Click **Display <-> Edit**.

3. Do one of the following:
   - To add a new field, insert an **ObjectValueSelector** from the **Toolbox** controls.
   - To modify an existing field, select the required field, for example Employee. In the **Properties Explorer** under Appearance, set **DisplayType** to **ObjectValueSelector**.

4. In the **Properties Explorer**, under **Value Help - OVSComponent**, click ...

   The **Select OVS Component** dialog is displayed.

5. Select the appropriate OVS dialog box component as the **Target Component**.

6. Click **OK**.

7. Bind the ID attribute to the value. The CCTS type must be the same as defined in the outport of the OVS dialog box.

8. Bind the description attribute to the text. The CCTS type must be the same as defined in the outport of the OVS dialog box. The value for the text is not passed on automatically from the OVS box. You use the **AfterModify** script files to ensure that the texts are retrieved.

9. Save and activate your changes.
Example
You are creating a sales order and want to assign an employee responsible for that order. In the Sales Order floorplan, there is an Employee field, which is an OVS control. If you click this control, the OVS dialog appears where you can search for the required employee. The selected employee is displayed in the Sales Order floorplan.

7.4.3.19 Configure a Dedicated Field

Overview
You can add a dedicated field to a data model and use a field transformation to calculate the value of this dedicated field from existing business object (BO) node attributes.

For more information, see, Transformations [page 535].

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. On the DataModel tab, right-click on Root and add a dedicated field.
4. On the Controller tab, right-click on Field Transformations and add a new Field Transformation.
5. For the Field Transformation, under Target Field, select the dedicated field that you created.
6. Select the Namespace.
7. Under Transformation, select the type of transformation rule you want to use. The parameters for that transformation rule are displayed. Each parameter section displays the parameter name and an indication if it is a mandatory field.
8. Each parameter can be configured by creating a data binding to the BO model or by assigning a constant value to it.
   - To configure a data binding, do the following:
     a. Select the BO binding checkbox.
     b. In the BO configuration area, click ....
     c. In the Select Data Field dialog box, select the BO node to be bound.
     d. Click OK.
   - To assign a constant value, select the Constant checkbox and, in the Constant field, enter a value.
9. Save and activate your changes.

7.4.3.20 Configure A Default Set Parameter Using Transformations

Overview
You can use a transformation to calculate the value of a default set parameter. Different default sets are listed under a query.
For example, in the Contracts OWL, the Show drop down list has entries such as Contracts in Preparation, and Released Contracts. These are the default sets under a query and are used to display a filtered set of contracts. You can create a query and default sets under that query. These default sets are then added to the drop down list.

For more information, see, Transformations [page 535].

You can have two types of transformations to calculate default set parameters:

- An example of the first type would be a transformation where no input parameters are needed. The transformation calculates the value needed using the login ID (sy-uname) and the identity UUID is the output.
- An example for the second type would be a transformation where the input parameters can only be constants 0 or 1 and the output is placed in the Low Value field of the Default set parameter row.

Procedure

1. In the SDK, double-click the required floorplan in the Solution Explorer.  
The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. Add a data field, as follows:
   a. On the Data Model tab, right-click on Root and then add a data structure.
   b. Right-click on the structure and then add a data field.
4. Configure a query, as follows:
   a. On the Controller tab, right-click on Queries and then choose Add Query.
   b. In the Query Configuration, select the required Namespace and BO.
      The queries modeled under the BO are displayed in a hierarchical manner. Expand the query node in the tree to show all the query parameters.
   c. Select the required query and click Bind.
   d. Add the query Parameters and select the data field that you created as the required Value.
5. Configure the default set for the query, as follows:
   a. Right-click on the query and then choose Add Default Set.
   b. To rename the default set, right-click on the default set and then choose Rename.
   c. In the Default Set Configuration, enter a Title for the default set. This value is displayed in the Show field.
   d. Select the query parameter that you have added and click Add Option.
   e. In the new row that appears, under Transformation click …
   f. In the Configure Field Transformation dialog box, select the Namespace.
   g. Under Transformation, select the type of transformation rule you want to use.
      The parameters for that transformation rule are displayed. Each parameter section displays the parameter name and an indication if it is a mandatory field.
      Each parameter can be configured by assigning a constant value to it.
      For some transformation rules, there is no provision for entering input parameters.
   h. In the Constant field, enter the value of the constant.
6. Save and activate your changes.
7.4.3.21 Add a Report List View to a Work Center

Overview

You need to assign standard report list views, for example the List and Gallery standard reports list views, to a work center in the User Interface Designer.

In the customer’s production system of the SAP solution, an Analytics key user assigns the report to the work center view to make it available for business users.

Procedure

1. In the SDK, double-click the required work center in the Solution Explorer. The work center opens in the User Interface Designer.
2. Click Display <-> Edit.
3. On the Designer tab, right-click on BusinessObject and select WorkCenterItem.
4. In the Properties Explorer, under Misc-ViewSwitches click ….
5. In the ViewSwitch Collection Editor dialog box, do the following:
   a. Click Add to add a new view.
   b. Select the newly added view.
   c. In the Properties Explorer, under Misc-SubViewSwitches, click ….
      The SubViewSwitch Collection Editor dialog box opens.
6. In the SubViewSwitch Collection Editor dialog box do the following:
   a. Click Add twice to add two new subviews.
   b. Click OK.
7. In the ViewSwitch Collection Editor dialog box, click OK.
8. On the Designer tab, select WorkCenterView.
9. Under WorkCenterView Views, select the first WorkCenterSubView that you added.
10. In the Configuration Explorer, navigate to SAP_BYD_TF Analytics AnalysisPattern ana_reports_view_wc.WCVIEW.uiwocview.
11. Drag and drop this work center view onto the Designer tab.
12. In the dialog box that opens, click Yes to overwrite the work center view floorplan title with the view switch name.
13. Under WorkCenterView Views, select the second WorkCenterSubView that you added.
14. In the Configuration Explorer, navigate to SAP_BYD_TF Analytics AnalysisPattern ana_reports_carousel_view.WCVIEW.uiwocview.
15. Drag and drop this work center view onto the Designer tab.
16. In the dialog box that opens, click Yes to overwrite the work center view floorplan title with the view switch name.
17. On the Designer tab, select WorkCenterView.
18. In the Properties Explorer, under Misc-Text click the arrow button.
19. In the Dependent Property Editor dialog box, do the following:
   a. Change the text in the Overridden Text field to Reports.
   b. Click OK.
20. Save and activate your changes.

See Also
Create a Report  [page 341]

7.4.3.22 Define an Anchor for a Floorplan

Overview
You can define anchors for your floorplans to allow end users to make changes to the floorplans provided by you.

Procedure
1. In the SDK, double-click the required floorplan in the Solution Explorer. The floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. Select the region in floorplan for which you want to define the anchor.
   For example, you can define the anchor for the floorplan as a whole or for a section group, pane container, outport, and so on.
4. In the Properties Explorer, under Extensibility–Anchors, click ....
5. In the Anchor Modelling dialog box, click + to add an anchor.
6. Click OK.
7. Save and activate your changes.

   You can view the anchors you have added only in the display mode.

7.4.3.23 Map a Message

Overview
You can map a message in order to change the terminology in messages appearing on the UI or to change their behavior on the UI. For more information see Message Mapping  [page 538].
Procedure

1. In the Solution Explorer, click on the UI component and choose Open in UI Designer. The UI Designer opens with the selected UI component in a new window.

2. Change to edit mode, go to the Controller tab, and select Messages –> System Messages. The Messages Editor opens.


4. For the new message text that you want to be displayed on the UI instead of the original text do the following:
   a. In the Message Key column enter a key for the new message, instead of <MESSAGETEX>. The message key consists of alphanumeric characters.
   b. In the Message Text column enter the new message text for <message text>.

5. Save and activate your changes.


7. Select Add Messages and in the first free line of the Messages Editor select ... to look for the message that you want to replace on the UI. A new window opens listing all the messages that are associated with the UI component.

8. Select the message that you want to replace and choose OK. The window closes and the message that you want to replace is displayed as a new line in the Message Mappings Editor.

9. In the Message Mappings Editor, select the rule that you want to apply in your message mapping.
   - To replace an original message with a new message, select Replace.
   - To keep the original message without any change, select Original.
   - To combine two or more original messages into one new single message, select Aggregate for each original message and select the same message key for each of them in the field New Message Key.
   - To prevent a message from being shown on the UI during runtime, select Ignore.

10. To show different mapped messages based on parameter values in one original message, proceed as follows: In the Details section of the Message Mappings Editor, in the field Message Parameter, select the parameter and in the field Parameter Value, enter the parameter value that you want the mapping to be based on.

11. On the Details screen in the field New Message Text, select the message text that you want to appear instead of the original message text.

If you want to aggregate two or more original messages, make sure that always select the same new message for those original messages.

12. Save and activate your changes.
7.5 Modify an SAP Floorplan

7.5.1 Change Transactions Quick Guide

This document gives you an overview of the changes you can make to SAP floorplans in the SDK. These changes are termed as change transactions and need anchors to be executed.

Business and technical background

Change Transactions
Change transactions allow you to make changes to SAP delivered floorplans in the SDK. You can create, save, and publish the change transactions using the Extensibility Explorer of the User Interface Designer, which is integrated into the SDK.

Change transactions can only be made to SAP floorplans that contain stable anchors. For more information, see Anchors [page 564].

For more information, see here [page 564].

Anchors
Anchors are stable extension points to which any user interface modifications that you make to SAP floorplans are attached. This prevents life-cycle issues during updates or upgrades to the user interface. You can also add anchors to your own floorplans.

For more information, see here [page 564].

Tasks

Add a View to a Work Center
You can add a view to an SAP work center.
For more information, see here [page 565].

Hide a View from a Work Center
You can hide a view that is no longer required from an SAP work center.
For more information, see here [page 566].

Add an Assigned Object to a View
You can assign an object, for example, a navigation target or user interface component, to an SAP work center view.
For more information, see here [page 567].

Remove an Assigned Object from a View
You can remove unwanted objects from an SAP work center view.
For more information, see here [page 567].
Add an Embedded Component to an Overview View
You can add an embedded component such as an embedded report (ICP) to an SAP work center view of type "Overview".
For more information, see here [page 568].

Add an Embedded Component to a Floorplan
You can add an embedded component to an SAP floorplan.
For more information, see here [page 569].

Add an Embedded Component to a New Tab in a Floorplan
You can add an embedded component to a new tab in an SAP floorplan.
For more information, see here [page 570].

Add an Extension Field to a Screen
You can add an extension field to a screen. For more information see here [page 348].

Add an Extension Field to the Enterprise Search
You can add an extension field to the enterprise search. For more information see here [page 352].

Remove a Button
You can remove a button or navigational item from an SAP floorplan.
For more information, see here [page 571].

Modify the Order of Work Centers
You can modify the order in which the work centers are displayed in the UI Settings XML file.
For more information, see here [page 571].

Add a User Interface Dependency
User Interface (UI) dependencies govern the assignment of views to a user. Three types of dependencies between views can be maintained.
For more information, see here [page 572].

Save and Activate your Change Transactions
After you have made any of the changes listed above, you have to save and activate the change transactions.

1. Make your changes and then click **Apply/Add**.
   A list of unsaved change transactions is displayed in the **Unsaved Change Transactions** section of the **Extensibility Explorer**.
2. From the **File** menu, choose **Save**.
   A list of saved change transactions is displayed in the **Saved Change Transactions** section of the **Extensibility Explorer**.
3. From the **File** menu, choose **Activate**.
   In the **Activate Worklist** dialog box, select the check boxes beside the relevant components and click **OK**.
   The change transactions are activated and displayed in the **Published Change Transactions** section of the **Extensibility Explorer**.
7.5.2 Business and technical background

7.5.2.1 Change Transactions

**Overview**

Change transactions allow you to make changes to SAP delivered floorplans in the SDK. You can create, save, and publish the change transactions using the Extensibility Explorer of the User Interface Designer, which is integrated into the SDK.

Change transactions can only be made to SAP floorplans that contain stable anchors. For more information, see Anchors [page 564].

When you create change transactions they are stored in the associated solutions created by you.

You can use a change transaction to add a view to a work center delivered by SAP. For example, you can add a new view to the Liquidity Management work center (CashLiquidityManagement_WCF).

7.5.2.2 Anchors

**Overview**

Anchors are stable extension points to which any user interface modifications that you make to SAP floorplans are attached. This prevents life-cycle issues during updates or upgrades to the user interface. You can also add anchors to your own floorplans.

Any user interface adaptations made by you, as an SAP partner, to SAP floorplans, are retained unchanged in future releases even after the upgrades provided by SAP due to the presence of anchors.

**Features**

A stable anchor has a system-wide unique ID and a reference entity. All stable anchors created during a particular release will also exist in later releases.

Anchors are essential when making changes to a floorplan that is from a layer lower than the layer you are working on. You can only view the anchors added in a lower layer. For example, when you are working on a SAP floorplan you can view the anchors provided by SAP. You can view the anchors added by you only when a floorplan is opened in the display mode.

Stable anchors in the floorplans are defined for a logical set of fields placed on the user interface, for example a section group or a list.

**Use**

All change transactions can only be applied if there is a stable anchor defined on the SAP floorplan.
### See Also

* Define an Anchor for a Floorplan  [page 560] *

### 7.5.3 Tasks

#### 7.5.3.1 Add a View to A Work Center

### Overview

You can add a view to an SAP work center.

### Prerequisites

The SAP work center to which you want to add a view has a `WoCViewSwitchAnchor`. For more information, see Anchors  [page 564].
Procedure

1. In the SDK, open the User Interface Designer.
2. In the Configuration Explorer, navigate to the work center where you want to add the view.
3. Double-click the work center.
4. In the Extensibility Explorer, select WoCViewSwitchAnchor.
5. Click Add and then choose AddWoCView.
6. In the Add Work Center View dialog box, click Target Component to select the required view. You can only select components of type uiwocview (work center view) and subtype Application.
7. Select the required position:
   - Top places the new view as the first view in the work center. If an Overview view exists in the work center, then the new view is positioned after the Overview view.
   - Bottom places the new view as the last view in the work center. If a Reports view exists, then the new view is positioned before the Reports view.
8. If you want to control the scope dependency of the view in the work center, click Add Scoping Rules.
9. In the Scope Rules Editor dialog box, choose the scoping rule from the list on the left and use the arrow buttons to move it to the list on the right.
10. Click OK, to apply your changes to the view.
11. Click Apply.
12. Save and activate your changes.

7.5.3.2 Hide a View from a Work Center

Overview

You can hide a view that is no longer required from an SAP work center.

Prerequisites

- The work center has a WoCViewSwitchAnchor. For more information, see Anchors [page 564].

Procedure

1. In the SDK, open the User Interface Designer.
2. In the Configuration Explorer, navigate to the work center in which you want to hide a view.
3. Double-click the work center.
4. In the Extensibility Explorer, select WoCViewSwitchAnchor.
5. Click Add and then choose HideWoCView.
6. In the Hide Work Center View dialog box, select the title of the view you want to hide.
7. Click Apply.
8. Save and activate your changes.
7.5.3.3 Add an Assigned Object to a View

Overview
You can assign an object, for example, a navigation target or user interface component, to an SAP work center view.

Prerequisites
The SAP work center view, to which you want to assign an object, has an AssignedObjectAnchor. For more information see, Anchors [page 564].

Procedure
1. In the SDK, open the User Interface Designer.
2. In the Configuration Explorer, navigate to the work center view where you want to assign the object.
3. Double-click the work center view.
4. In the Extensibility Explorer, select AssignedObjectAnchor.
5. Click Add and then choose AddAssignedObjectToView.
6. In the Add Assigned Object dialog box, select the type of object you want to assign.
7. Click Target Component to select the object.
8. Depending on the type of assigned object, you have the following options:
   - If the assigned object is of type Navigation Target, you can add scoping rules:
     1. Click Add Scoping Rules.
     2. In the Scope Rules Editor dialog box, choose the scoping rule from the list on the left.
     3. Use the arrow buttons to move the selected scoping rule to the list on the right. Click OK to apply your changes.
   - If the assigned object is an object work list, then you can mark the view as Expose to Overview. The object work list will then be visible in the Overview view.
9. Click Apply.
10. Save and activate your changes.

7.5.3.4 Remove an Assigned Object from a View

Overview
You can remove unwanted objects from an SAP work center view.

Prerequisites
The work center view has an AssignedObjectAnchor. For more information, see Anchors [page 564].
Procedure

1. In the SDK, open the User Interface Designer.
2. In the Configuration Explorer, navigate to the work center view in which you want to remove an assigned object.
3. Double-click the work center view.
4. In the Extensibility Explorer, select AssignedObjectAnchor.
5. Click Add and then choose RemoveAssignedObjectToView.
6. In the Remove Assigned Object dialog box, select the object you want to remove.
7. Click Apply.
8. Save and activate your changes.

7.5.3.5 Add an Embedded Component to an Overview View

Overview

You can add an embedded component such as an embedded report (ICP) to an SAP work center view of type "Overview".

Procedure

1. In the SDK, open the User Interface Designer.
2. In the Configuration Explorer, navigate to the Overview view where you want to add an embedded component.
3. Double-click the floorplan.
4. In the Extensibility Explorer, select the OverviewPageAnchor.
5. Click Add and then choose AddContenttoOverview.
6. In the Add Component dialog box, do the following:
   a. Select the floorplan type, for example, an object work list.
   b. Click ... under Component and select the floorplan.
   c. Select the required Title, Position and Alignment of the embedded component.
7. If the embedded component is of the generic type ICP, you have to select the parameters associated with the ICP under the option View.
8. Click Add.
9. Save and activate your changes.
7.5.3.6 Add an Embedded Component to a Floorplan

Overview

You can add an embedded component to an SAP floorplan. You can add the embedded component to either an OutportAnchor or a PaneContainerAnchor.

Prerequisites

- The floorplan has an OutportAnchor.
  For more information, see Anchors [page 564].
- The floorplan has a PaneContainerAnchor.
  The embedded component (EC) will be placed behind the pane container.
- The embedded component has an inport.
  The RequestAutoRefire and RequestFireOnInitialization properties of the embedded component are set to True.

Procedure

1. In the SDK, open the User Interface Designer.
2. In the Configuration Explorer, navigate to the floorplan in which you want to add an embedded component.
3. Double-click the floorplan.
4. In the Extensibility Explorer, do one of the following:
   - To add the embedded component to an OutportAnchor:
     a. Select the OutportAnchor, click Add and then choose Add Embedded Component.

       The outport selected has an OutportAnchor and the PaneContainerAnchor set as the referenced anchor.

     b. In the Add Embedded Component dialog box, click Select Embedded Component and select the required embedded component.
     c. In the Add Embedded Component dialog box, if the referenced pane container for which the PaneContainerAnchor has been defined, contains a tab strip, you can enter a Tab Title.

       1. At runtime, a new tab is created with this title, and the embedded component appears in the tab.
       2. If the referenced pane container does not contain a tab strip, the embedded component appears in a new row below the referenced pane container.

     d. Click Define Mapping and select the inport of the embedded component.
     e. If the inport has any parameters, map the parameters.
     f. If the embedded component is of the generic type ICP, you are prompted to provide a title for the ICP.

   - To add the embedded component to a PaneContainerAnchor:
     a. Select the PaneContainerAnchor, click Add and then choose Add Embedded Component.
b. In the Add Embedded Component dialog box, if the referenced pane container, for which the PaneContainerAnchor has been defined, contains a tab strip, you can enter a Tab Title.
   1. At runtime, a new tab is created with this title, and the embedded component appears in the tab.
   2. If the referenced pane container does not contain a tab strip, the embedded component appears in a new row below the referenced pane container.

c. Click Select Embedded Component and select the required embedded component.

d. If you want to bind the embedded component to the floorplan, click Bind.

e. In the dialog box that opens, select the outport and inport for the embedded component. The outport selected has an OutportAnchor.

5. Click Apply.
6. Save and activate your changes.

7.5.3.7 Add an Embedded Component to a New Tab in a Floorplan

Overview

You can add an embedded component to a new tab in an SAP floorplan.

Prerequisites

- The floorplan has either the ViewSwitchNavigationAnchor or the TabStripAnchor. For more information, see Anchors [page 564].
- The embedded component has an inport.

Procedure

1. In the SDK, open the User Interface Designer.
2. In the Configuration Explorer, navigate to the floorplan in which you want to add an embedded component.
3. Double-click the floorplan.
4. In the Extensibility Explorer, select the ViewSwitchNavigationAnchor or the TabStripAnchor.
5. Click Add and then choose Add Embedded Component.
6. In the Add Embedded Component dialog box, enter a Tab Title.
7. Click Select Embedded Component and select the required embedded component.
8. If you want to bind the embedded component to the floorplan, click Bind.
9. In the dialog box that opens, select the outport and inport for the embedded component. The outport selected has an OutportAnchor associated with it.
10. Click Apply.
11. Save and activate your changes.
7.5.3.8 Remove a button

**Overview**
You can remove a button or navigational item from an SAP floorplan.

**Prerequisites**
The SAP floorplan from which you want to remove a button or navigational item has a `ButtonGroupAnchor`. For more information, see Anchors  [page 564].

**Procedure**
1. In the SDK, open the User Interface Designer.
2. In the Configuration Explorer navigate to the required floorplan.
3. Double-click the floorplan.
4. Select a button group in the floorplan.
5. In the Extensibility Explorer, select the `ButtonGroupAnchor`.
6. Click Add and then choose `RemoveButton`.
7. In the Remove Button dialog box, select a button or navigational item that you want to remove from the floorplan.
   If the button or navigational item cannot be removed, the Apply button is disabled.
   The removal of mandatory buttons like Cancel, Close, and Print is restricted.
8. Click Apply.
9. Save and activate your changes.

7.5.3.9 Modify the Order of Work Centers

**Overview**
You can modify the order in which the work centers are displayed in the UI Settings XML file, the path to which is given below.

**Prerequisites**
The UI Settings file has a `WorkCenterListAnchor`. For more information, see Anchors  [page 564].
Procedure

1. In the SDK, open the User Interface Designer.
2. In the Configuration Explorer, go to \[SAP_BYD_UI_CCOW\] Main root uisetting.
3. Double-click the file.
4. In the Extensibility Explorer, select the WorkCenterListAnchor.
5. Click Add and then choose WorkCenterReordering.
6. You can do the following in the Work Center Reordering dialog box:
   - Add work centers to the list of Ordered WorkCenters.
     ○ Select the required work center from the list of Unordered WorkCenters and use the right arrow button to add the work center to the list of Ordered WorkCenters.
     The required work center is added to the list of Ordered WorkCenters, below a work center selected in the list.
   - Remove work centers from the list of Ordered WorkCenters.
     ○ Select the required work center from the list of Ordered WorkCenters and use the left arrow button to move the work center to the list of Unordered WorkCenters.
   - Change the position of the work center.
     ○ Select a work center and use the up and down arrow buttons to change its position in the list of Ordered WorkCenters.
     ○ To place the selected work center at the top of the list of Ordered WorkCenters click Top.
     ○ To place the selected work center at the bottom of the list of Ordered WorkCenters click Bottom.
7. Click Apply.
8. Save and activate your changes.

7.5.3.10 Add a User Interface Dependency

Overview

User interface (UI) dependencies govern the assignment of views to a user. Three types of dependencies between views can be maintained.

Types of Dependencies

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisite</td>
<td>Ensures that if one work center view is assigned to a user, then the other work center view in the dependency is automatically assigned to the user.</td>
</tr>
<tr>
<td>Technical Conflict</td>
<td>Ensures that the user is not assigned two work center views that conflict with each other from a technical point of view. When there is a technical conflict, an error message is displayed and assignment of views is not allowed. For example, if both the Postings and Master Data views of the Fixed Assets workcenter are assigned to a user, then an error message appears and the assignment is not carried out.</td>
</tr>
<tr>
<td>SoD Conflict (Segregation of Duty)</td>
<td>Ensures that the user is not assigned two views (chosen while defining the dependency) that conflict with each other from a best practice point of view. For example, if the user has two views assigned that enable the user to both place an order and to approve the order. These functions should be performed by different users. When a user assignment generates a SoD conflict, a warning message is displayed.</td>
</tr>
</tbody>
</table>
These user interface dependencies will govern the assignment of views in the User and Access management view. For more information, see User and Access Management.

The existing dependencies are listed in the UIDependency XML file, the path to which is given below.

Prerequisites

The UIDependency XML file has a UIDependencyAnchor.
For more information, see Anchors [page 564].

Procedure

1. In the SDK, open the User Interface Designer.
2. In the Configuration Explorer, go to ➤ SAP_BYD_UI_CCOW ➤ Main ➤ root ➤ uidependency ➤
3. Double-click the UIDependency XML file.
4. In the Extensibility Explorer, select the UIDependencyAnchor.
5. Click Add and then choose UI Dependency.
6. You can add three kinds of UI dependencies: Prerequisite, Technical Conflict, and SoD Conflict.
   • If you want to add a prerequisite dependency:
     a. In the UI Dependency dialog box, click the Prerequisites tab.
     b. Click + to add a new prerequisite.
     c. Select the desired views for Target Partner WoC View and Prerequisite WoC View.
   • If you want to add a technical conflict dependency:
     a. In the UI Dependency dialog box, click the Technical Conflict tab.
     b. Click + to add a new technical conflict.
     c. Select the desired views for Target Partner WoC View and Technical Conflicting WoC View.
     d. In the Reason field, enter the reason why a user should not be assigned both views at the same time.
   • If you want to add a SoD conflict dependency:
     a. In the UI Dependency dialog box, click the SoD Conflict tab.
     b. Click + to add a new SoD conflict.
     c. Select the desired views for Target Partner WoC View and SoD Conflicting WoC View.
     d. In the Reason field, enter the reason why a user should not be assigned both views at the same time.
     e. In the Resolution field, enter the solution to the issue.
7. Click Apply.
8. Save and activate your changes.
7.6 User Interface Modifications for Mobile Applications

7.6.1 Mobile Floorplans Quick Guide

This document gives you an overview of how you can adapt floorplans for use on mobile devices and the steps involved in configuring navigation between floorplans.

For more information on creating the floorplans, see Generate Screens for a Business Object [page 314].

Business and Technical background

SAP's User Interface Style Guide
You can learn how to build user interfaces according to this style guide.


Here you can also find a list of all current user experience rules. These are intended as a support for you to achieve a similar user experience with regard to interaction, look and feel as in the standard SAP solution. The rules are included in an automatic check tool inside the User Interface Designer.

Mobile Solutions
You can create solutions in the SDK and make them available on mobile devices, such as the iPhone and Blackberry. For more information, see Mobile Solutions.

Tasks

Modify an Object Work List for Use on a Mobile Device
You can modify an object work list (OWL) created for use on a mobile device.
For more information, see here [page 575].

Modify a Quick Activity for Use on a Mobile Device
You can modify a quick activity floorplan (QAF) created for use on a mobile device.
For more information, see here [page 576].

Modify a Factsheet for Use on a Mobile Device
You can modify an factsheet (FS) created for use on a mobile device.
For more information, see here [page 578].
Configure a Factsheet for Use on a Mobile Device
You can configure a factsheet (FS) so that you can navigate from an object work list (OWL) to a factsheet rather than navigating directly to a quick activity floorplan (QAF).
For more information, see here [page 579].

Assign a Mobile Factsheet to Work Center View
You can assign a mobile factsheet to a work center view to make it visible in the work center view.
For more information, see here [page 580].

Create a Port to Navigate to a Mobile Factsheet
For more information, see here [page 580].

Configure Navigation in a Mobile Factsheet
For more information, see here [page 581].

Configure Navigation in a Mobile Object Worklist
For more information, see here [page 581].

Configure Navigation Between a Mobile Factsheet and a Mobile Quick Activity
For more information, see here [page 583].

7.6.2 Tasks

7.6.2.1 Modify an Object Work List for Use on a Mobile Device

Overview
You can modify an object work list created for use on a mobile device, using the User Interface Designer, in the SDK.

Prerequisites

- You have created and activated the business object (BO) and other components in the SDK. For more information, see Business Objects Quick Guide [page 297].
- You have created the object work list in the SDK. For more information, see Generate Screens for a Business Object [page 314].

Procedure
1. In the SDK, double-click the object work list in the Solution Explorer. The object work list opens in the User Interface Designer.
2. Click Display <-> Edit.
3. In the Properties Explorer, under Configuration- Tags click ...

4. In the Tags Editor dialog box, do the following:
   a. Click Add Operation.
   b. In the new row that appears:
      i. Under Attributes, select Device.
      ii. Under Value, select Mobile.
   c. Click OK.

5. On the Designer tab, select the first column of the object work list.

6. In the Properties Explorer, do the following:
   a. Under Appearance- DisplayType, select Link.
   b. Under Events- OnClickEvent, select Edit Handler.

7. On the Designer tab, in the toolbar, delete all the buttons except the New button.
   \[\text{The New button is linked to the + of the object work list on the mobile screen.}\]

8. On the Designer tab, select the list.

9. In the Properties Explorer, under ChildElements- List columns click ...

10. In the Columns Collection Editor dialog box, select all columns except the first, click Remove and then, click OK.

11. In the dialog box that opens, click Yes to delete the data fields associated with the deleted columns.

12. In the Properties Explorer, under Appearance- ShowHeaderRow select False.

13. Save your changes.

\[\text{Note that, for the use on mobile devices, you have to activate the user interface elements in the following order:}\]
\begin{itemize}
  \item Work center
  \item Work center view
  \item Floorplans
\end{itemize}

7.6.2.2 Modify a Quick Activity for Use on a Mobile Device

\textbf{Overview}

You can modify a quick activity floorplan created for use on a mobile device, using the User Interface Designer, in the SDK.

\textbf{Prerequisites}

- You have created and activated the business object (BO) and other components in the SDK.
  For more information, see Business Objects Quick Guide [page 297].
- You have created the quick activity in the SDK.
  For more information, see Generate Screens for a Business Object [page 314].
Modify a Quick Activity of a Business Object without Actions for Use on a Mobile Device

1. In the SDK, double-click the quick activity floorplan in the Solution Explorer. The quick activity floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. In the Properties Explorer, under Configuration-Tags click ....
4. In the Tags Editor dialog box, do the following:
   a. Click Add Operation.
   b. In the new row that appears:
      • Under Attributes, select Device.
      • Under Value, select Mobile.
   c. Click OK.
5. On the Controller tab, under EventHandlers, rename the SaveHandler to SaveAndCloseHandler.
6. Under Configure the operation, change Action Type to SaveAndClose.
7. Save your changes.

Modify a Quick Activity of a Business Object with Actions for Use on a Mobile Device

1. In the SDK, double-click the quick activity floorplan in the Solution Explorer. The quick activity floorplan opens in the User Interface Designer.
2. Click Display <-> Edit.
3. In the Properties Explorer, under Configuration-Tags click ....
4. In the Tags Editor dialog box, do the following:
   a. Click Add Operation.
   b. In the new row that appears:
      • Under Attributes, select Device.
      • Under Value, select Mobile.
   c. Click OK.
5. On the Designer tab, in the toolbar, delete all the buttons except the Close button.
6. Select the Close button, and in the Properties Explorer do the following:
7. Select the contextual navigation region.
8. In the Properties Explorer, under Toolbar-Configure click ....
9. In the Configure Toolbar dialog box, do the following:
   a. In the Available list of buttons, under ApplicationSpecific buttons select My Button.
   b. Add My Button to the list of Current list of buttons using the right arrow button.
10. In the Properties Explorer, under MenuInformation-HasMenu select True.
11. In the Properties Explorer, under MenuInformation-SubMenu click ....
12. The Button SubMenu Collection Editor dialog box opens.
13. To add a new navigational item to the button sub menu, click Add.
14. For each action that you want to be displayed on the mobile device, add one navigational item.
15. Select the first navigational item and in the Properties Explorer do the following:
   a. Under Events-OnClick select SaveHandler.
   b. Under Design-Title enter the title of the button as Save.
16. Select the second navigational item and in the Properties Explorer do the following:
   a. Under Events-OnClick select Action<Name of Action in BO>Handler.
   b. Under Design-Title enter the title of the button as <Name of Action>.
17. Repeat steps 15 and 16 for all other actions of the business object that you want to use.
18. Click OK.
19. In the Properties Explorer, under Text Information-Text enter the text Actions.
20. Save your changes.

Note that, for the use on mobile devices, you have to activate the user interface elements in the following order:
   1. Work center
   2. Work center view
   3. Floorplans

7.6.2.3 Modify a Factsheet for Use on a Mobile Device

Overview
You can modify a factsheet created for use on a mobile device using the User Interface Designer in the SDK.

Prerequisites
- You have created and activated the business object (BO) and other components in the SDK. For more information, see Business Objects Quick Guide [page 297].
- You have created a factsheet in the SDK. For more information, see Generate Screens for a Business Object [page 314].

Procedure
1. In the SDK, double-click the factsheet in the Solution Explorer. The factsheet opens in the User Interface Designer.
2. Click Display <-> Edit.
3. In the Properties Explorer, under Configuration-Tags click ....
4. In the Tags Editor dialog box, do the following:
   a. Click Add Operation.
   b. In the new row that appears:
      • Under Attributes, select Device.
      • Under Value, select Mobile.
   c. Click OK.
5. On the Designer tab, delete the View All button.

6. Select the Close button, and in the Properties Explorer do the following:

7. Select the fields present in the pane container of the factsheet, and for each, in the Properties Explorer, under Appearance-DisplayType select Static text.

8. Delete the other pane containers in the factsheet.

9. Select the section group and do the following:
   a. Double-click the section group name.
   b. In the Dependent Property Editor dialog box, clear the text used for SectionGroupName.
   c. Click OK.

10. Select the section group and in the Properties Explorer, under Text Information-Show Header select False.

11. Delete the extended identification regions (ExIdr) from the factsheet.

12. Save your changes.

Note that, for the use on mobile devices, you have to activate the user interface elements in the following order:
   1. Work center
   2. Work center view
   3. Floorplans

7.6.2.4 Configure a Factsheet for Use on a Mobile Device

Overview
You can configure a factsheet for use on a mobile device. For example, you can configure a factsheet so that you can navigate from an object work list to a factsheet rather than navigating to a quick activity and then to a factsheet.

Prerequisites
- You have created and activated the business object (BO) and other components in the SDK.
  For more information, see Business Objects Quick Guide [page 297].
- You have created the following in the SDK:
  ○ Work center view
  ○ Port type package
  ○ Object work list
  ○ Quick activity floorplan
  ○ Factsheet
  For more information, see Generate Screens for a Business Object [page 314].
- You have modified the factsheet created for use on a mobile device, using the User Interface Designer, in the SDK.
  For more information, see Modify a Factsheet for use on a Mobile Device [page 578].
Process Flow

1. Assign the factsheet to a work center view.
   For more information, see Assign a Mobile Factsheet to a Work Center View  [page 580].

2. Create a port to allow the transfer of data between the factsheet and other floorplans.
   For more information, see Create a Port to Navigate to a Mobile Factsheet  [page 580].

3. Configure navigation in the mobile factsheet to allow the transfer of data between the fact sheet and the object work list.
   For more information, see Configure Navigation in a Mobile Factsheet  [page 581] and Configure Navigation in a Mobile Object Work List  [page 581].

4. Configure an event handler in the mobile factsheet.
   For more information, see Configure an Event Handler in a Mobile Factsheet  [page 582].

5. Configure navigation between the mobile factsheet and a mobile quick activity to allow the transfer of data from the quick activity to the factsheet.
   For more information, see Configure Navigation between a Mobile Factsheet and a Mobile Quick Activity  [page 583].

7.6.2.5 Assign a Mobile Factsheet to a Work Center View

Overview

You can assign a factsheet that has been modified for use on a mobile device to a work center view.

Procedure

1. In the SDK, double-click the work center view in the Solution Explorer.
   The work center view opens in the User Interface Designer.

2. Click Display <-> Edit.

3. In the Properties Explorer, under RBAM Data-AssignedObjects click ….

4. In the Add Assigned Objects dialog box, do the following:
   a. Click Add to add a new assigned object.
   b. In the new row, click … and select the factsheet you have created from the list of repository items.
   c. Click OK.

5. Save and activate your changes.

7.6.2.6 Create a Port to navigate to a Mobile Factsheet

Overview

You can create a new port to facilitate navigation to a factsheet for use on a mobile device.
Procedure

1. In the SDK, double-click the port type package in the Solution Explorer. The port type package opens in the User Interface Designer.
2. Click Display <-> Edit.
3. Right-click on the port type package and add a new port.
4. Rename the port as To_FS.
5. Save and activate your changes.

7.6.2.7 Configure Navigation in a Mobile Factsheet

Overview

You can configure navigation in a factsheet to facilitate the transfer of data between the factsheet and a object work list.

Procedure

1. In the SDK, double-click the mobile factsheet in the Solution Explorer. The factsheet opens in the User Interface Designer.
2. Click Display <-> Edit.
3. On the DataModel tab, right-click on Root and add a new data field.
4. Rename the data field as Node ID.
5. On the Controller tab, right-click on Inports and add a new inport.
6. Under Inport Configuration do the following:
   a. Select the port type package that you have created.
   b. Set the port type reference to the port To_FS.
   c. Add a parameter and bind it to the data field Node ID that you have created.
7. Save and activate your changes.

7.6.2.8 Configure Navigation In A Mobile Object Work List

Overview

You can configure navigation in a object work list to facilitate data exchange to a factsheet.

Procedure

1. In the SDK, double-click the object work list in the Solution Explorer. The object work list opens in the User Interface Designer.
2. Click Display <-> Edit.
3. On the **Controller** tab, right-click on **Outports** and add a new outport.
4. Rename it as **OutportToFS**.
5. Under **Outport Configuration** do the following:
   a. Select the port type package that you have created.
   b. Set the port type reference to the port **To_FS**.
   c. Add a parameter and bind it to the data field **Node ID**.
   
   ![](image)
   The parameter name should be the same in the factsheet and the object work list.
6. On the **Designer** tab, select the column in the advanced list pane.
7. In the **Properties Explorer**, under **Appearance-DisplayType** select **Link**.
8. In the **Properties Explorer**, under **Navigation Information-Navigation** click **…**.
9. In the **Model Navigation** dialog box, do the following and click **OK**:
   a. Click on **Simple navigation**.
   b. Under **Navigation Details**, set the target as the factsheet you have created.
   c. Under **Inport**, choose the inport you created in the factsheet.
   d. Under **Outport**, choose the outport you created in the object work list.
   e. Select the parameters under **Outport Parameters** and **Inport Parameters** and click **Bind**.
10. Save and activate your changes.

### 7.6.2.9 Configure an Event Handler in a Mobile Factsheet

**Overview**

You can configure an event handler in a factsheet to display data from a quick activity.

**Procedure**

1. In the SDK, double-click the factsheet in the **Solution Explorer**.
   The factsheet opens in the User Interface Designer.
2. Click **Display <-> Edit**.
3. On the **Controller** tab, right-click on **EventHandlers** and add a new event handler.
4. Rename the event handler as **ReadHandler**.
5. For the **ReadHandler**, under **ReadHandler : Operations**, select the operation type **BO Operation**.
6. Set the **BO Operation** as **Read**.
7. Add a parameter, select **nodeId** as parameter type and bind it to the data field **NodeId**.
8. Bind the path of the parameter to the data field **NodeId**.
9. Select the inport you have created in the factsheet.
10. In the **Properties Explorer**, under **Events-OnFire** select **Readhandler**.
11. Save and activate your changes.
7.6.2.10 Configure Navigation Between A Mobile Factsheet and A Mobile Quick Activity

Overview
You can configure navigation between a factsheet and a quick activity so that it is possible to transfer of data between them.

Procedure
1. In the SDK, double-click the factsheet in the Solution Explorer. The factsheet opens in the User Interface Designer.
2. Click Display <-> Edit.
3. On the Designer tab, select the Edit button.
4. In the Properties Explorer, under Misc-Name, rename the button as EditEH.
5. On the Controller tab, right-click on Outports and add a new outport.
6. Under Outport Configuration do the following:
   a. Select the port type package that you have created.
   b. Set the port type reference to the port To_<Floorplanname>_edit.
   c. Add a parameter and bind it to the data field Node ID.
7. On the Designer tab, select the Edit button.
8. In the Properties Explorer, under MenuInformation-Navigation click ....
9. In the Model Navigation dialog box, do the following and click OK:
   a. Click on Simple navigation.
   b. Under Navigation Details, set the target as the quick activity you have created.
   c. Under Import, choose the import OBNEditImport.
   d. Under Outport, choose the outport you created in the factsheet.
   e. Change Style to inplace.
   f. Select the parameters under Outport Parameters and Import Parameters and click Bind.
10. On the Controller tab, rename the event handler generated by the simple navigation to EditEH.
11. Save and activate your changes.
8 Style Guides and Reference Guides

8.1 Overview of Style Guides and Reference Guides

Style Guides and Reference Guides encompasses the following:

<table>
<thead>
<tr>
<th>Style Guides</th>
<th>Reference Guides</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Style Guide for Writing System Message Texts</strong> [page 584]</td>
<td><strong>Reference Guide for SAP Solution Documentation</strong> [page 609]</td>
</tr>
<tr>
<td>This style guide provides you with the standards and guidelines for writing system messages to be displayed on the user interface (UI) of your solution. It also provides the standards and guidelines for writing system messages that are relevant for SAP Support; these messages are not displayed on the UI. In addition, this style guide provides guidance on writing long-text documents providing additional information to a system message that helps the user to solve very complex error situations. These long-text documents can be incorporated into the Solution Documentation.</td>
<td>This reference guide provides an overview of the main reference documentation types specific to the SAP Business ByDesign solution.</td>
</tr>
</tbody>
</table>

8.2 Style Guides

8.2.1 Style Guide for Writing System Message Texts

8.2.1.1 General Standards and Guidelines for Writing System Message Texts

This style guide provides you with the standards and guidelines for writing system messages to be displayed on the user interface (UI) of your solution. It also provides the standards and guidelines for writing system messages that are relevant for SAP Support; these are not displayed on the UI.
In addition, this style guide provides guidance on writing long-text documents providing additional information to a system message that helps the user to solve very complex error situations. These long-text documents can be incorporated into the Solution Documentation.

**Target Group**

This style guide is aimed at solution partners and reseller partners, who develop solutions for SAP Business ByDesign using the software development kit (SDK).

**Prerequisites**

You are familiar with the following documents:
Types of System Message Texts

System messages in the SAP solution comprise the following message types:

- System message short texts
  - UI system message short texts
    System messages written directly for the UI address the end user directly. They provide concrete instructions on how to solve an error on the UI and contain standard SAP solution terminology.
  - Other system message short texts
    System messages written for developers and SAP Support. These messages are not aimed at the end user; they can be quite technical, describing the error situation and the reasons why an error occurred.

- Long-text documents with additional help to a UI system message short text.
  For very complex error situations, where the user needs to be provided with detailed background information about the error, a document can be written and provided as part of the solution documentation.

For information about how and where system messages appear on the UI, see the User Interface Style Guide.

8.2.1.2 General Standards for Message Short Texts

Overview

This section of the style guide provides you with the general standards and guidelines for writing message short texts.

Standard formulations for different categories of message short texts are contained in Standard Formulations for Message Short Texts [page 588].

Content

1. Diagnose and describe the error.
   - For technical reasons, system messages are restricted to 73 characters. If there is not enough space, for very complex error situations, where the user needs to be provided with detailed background information about the error, a long text can be written and provided as part of the solution documentation. For more information, see the section on General Standards for Message Long Texts [page 602].

2. If possible, provide a reason for the error.

3. If required, provide instructions on how to solve the error.
Sentence Construction

1. Formulate simple messages that are easily understandable.
2. Use precise and unequivocal language. Provide a maximum of information in a minimum of space.
   - Correct: Recipient supply planning area not active
   - Incorrect: Recipient supply planning area has wrong status
     Note: It is more precise to name the correct status.
3. Use sentence style.
   Some of the messages that appear in application logs deviate from this standard. They are used as headings (title case). For more information, see the section on Standards for Application Log Messages [page 597].
4. Do not use a period or exclamation mark at the end of a message short text.
5. If the message text consists of two parts, separate them using a semicolon (;)
   - Correct: Error during RFQ creation; submission deadline missing
   - Incorrect: Error During RFQ Creation! Submission Deadline Missing.
6. Do not use an article at the start of a message.
   - Correct: Hierarchy &1 already exists
   - Incorrect: The hierarchy &1 already exists
7. Omit articles within the text unless this creates ambiguity.
   - Correct: Buyer not authorized to display master data
   - Incorrect: Buyer not authorized to display the master data
8. Use the tense that is appropriate to the context.
   - Correct: Payment order creation failed
   - Incorrect: Payment order creation fails
   - Incorrect: Payment order creation will fail

Word Choice

1. Use terminology from the Public Solution Model for system messages that do not appear on the UI.
2. Use terminology from the SAP solution for UI messages.

- Correct: Sourcing request currency missing
- Incorrect: RFQ request currency missing

Note: The business object is named RFQ Request. On the UI the term used is Sourcing Request.

3. Avoid abbreviations wherever possible.

4. Do not use generic modeling terms such as type code.

- Correct: Operation of type &1 not permitted
- Incorrect: Operation category type code &1 not permitted

5. Do not use the word please.

- Correct: Activate the document
- Incorrect: Please activate the document

Special Characters and Placeholders

1. Do not use special formatting, for example, upper case letters for emphasis. In some languages, the use of upper case can be construed as shouting.

- Correct: Error during customer invoice creation
- Incorrect: ERROR during customer invoice creation

2. Do not use quotation marks ("...") to set off placeholders, statuses, or similar.

3. Do not use special characters such as angle brackets (&lt;...&gt;), dollar signs ($), or asterisks (*).

4. Use ampersands (&) only for variables. If you use more than one variable, number your variables (&1, &2, &3, &4).

5. Do not write a message text that only consists of variables (for example, &1 &2 &3). Bear in mind that different languages have different word order. A message short text that reads correctly in the source language may be nonsensical in another language.

6. If a noun is preceded by a variable that is a placeholder for a cardinal number, do not set off the plural form or the plural ending of noun and verb with parentheses (( ... )) or a slash (/). Use only the plural form.

Adding the plural endings in parentheses or with a slash does not work with irregular plural forms and with every language. Using the plural form may sometimes result in incorrect sentences, such as 1 documents not available, but in the majority of cases the variable will be replaced with a number higher than 1.
7. In general, avoid using formulations that cause problems in translation, especially in texts containing placeholders and ordinal numbers.

- Correct: Deletion of quantities not possible
- Correct: Line &1 deleted
  Note: This message is displayed on the UI as, for example, Line 22 deleted.
- Correct: Enter a valid quantity
- Correct: Enter a valid value in the &1 field
  Note: In this example, &1 stands for the Quantity field label (UI text). This message is displayed on the UI as Enter a valid value in the Quantity field.
- Incorrect: Cannot &1 quantities
  Note: Do not use a placeholder for a verb.
- Incorrect: &1th line deleted
  Note: Do not use a placeholder for an ordinal number. This message is displayed on the UI as 22th line deleted, which is incorrect in English.
- Incorrect: Enter a valid &1
  Note: In this example, &1 stands for the Quantity field label (UI text). This message is displayed on the UI as Enter a valid Quantity. The capitalization is incorrect in English, and the adjective cannot be translated correctly into another language.

8.2.1.3 Standard Formulations for Message Short Texts

Overview

This section of the Style Guide and its subsections provides standard formulations for the most common message short texts. They are listed and structured according to the error category to which they belong. Whenever you write a message, analyze the error situation and decide to which category it belongs. If none of the categories listed here fits the error situation of your message, follow the General Standards for Message Short Texts [page 585] to write the message.

Error Categories

<table>
<thead>
<tr>
<th>Errors and Failures</th>
<th>• Failure [page 589]</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Errors [page 589]</td>
<td></td>
</tr>
<tr>
<td>• Errors Due to Business Configuration [page 590]</td>
<td></td>
</tr>
<tr>
<td>Inability to Perform Action, Status and Authorization Problems</td>
<td>• Inability to Perform Action [page 590]</td>
</tr>
<tr>
<td>• Inability to Perform Action Because Object is Locked [page 591]</td>
<td></td>
</tr>
<tr>
<td>• Inability to Display Data [page 591]</td>
<td></td>
</tr>
<tr>
<td>• Lack of Authorization [page 591]</td>
<td></td>
</tr>
<tr>
<td>• Inappropriate Status [page 592]</td>
<td></td>
</tr>
<tr>
<td>• Mandatory Step [page 592]</td>
<td></td>
</tr>
</tbody>
</table>
8.2.1.4 Standards for Errors and Failures

8.2.1.4.1 Failure — Standard Formulations

You use this standard formulation in cases where the execution of an action fails completely or where a general failure occurs.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>... failed; &lt;instruction or reason&gt;</td>
<td>Printing failed; install printer</td>
<td>Error during printing</td>
</tr>
<tr>
<td></td>
<td>Printing failed; printer settings not correct</td>
<td>Printing had a failure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Internal error; internal data inconsistency</td>
</tr>
</tbody>
</table>

Printing failed means that a document has not been printed at all.

8.2.1.4.2 Errors — Standard Formulations

You use these standard formulations in cases where the execution of an action has not failed completely, but can only be completed with errors.
8.2.1.4.3 Errors Due to Business Configuration— Standard Formulations

You use these standard formulations in cases where an error occurs due to wrong or missing business configuration.

If possible, provide detailed information about what is wrong in the business configuration in a message long text. For more information, see General Standards for Message Long Texts [page 602].

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error in business configuration for ...</td>
<td>Error in business configuration for purchasing contract</td>
<td>Business configuration error: Maintain purchasing contract</td>
</tr>
<tr>
<td></td>
<td>Error in business configuration for pricing procedure</td>
<td>Pricing not maintained in business configuration</td>
</tr>
<tr>
<td>Note: For the message long text, an example is The system could not determine or assign the Catalog for Pricing field. ...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.2.1.5 Standards for Inability to Perform Action, Status and Authorization Problems

8.2.1.5.1 Inability to Perform Action — Standard Formulations

You use these standard formulations in cases where an action cannot be performed or is not permitted. If the action is not followed by an object, use the technical action name, for example, Delete, Activate, and so on.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Action name&gt; not permitted; &lt;instruction or reason&gt;</td>
<td>Delete not permitted for user status; user not authorized to delete objects</td>
<td>Function prohibited for user status</td>
</tr>
<tr>
<td>&lt;Action name&gt; not possible; &lt;instruction or reason&gt;</td>
<td>Create not possible; source object does not exist</td>
<td>Creation not possible, because source object does not exist</td>
</tr>
</tbody>
</table>
If the action is followed by an object, you may use the gerund or another verbal noun, for example, *Deleting the document*, and so on.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>... &lt;object&gt; not permitted; &lt;instruction or reason&gt;</td>
<td>Deletion of quantities not permitted; user not authorized to delete quantities</td>
<td>Delete quantities not allowed; user not authorized to delete quantities</td>
</tr>
<tr>
<td>... &lt;object&gt; not possible; &lt;instruction or reason&gt;</td>
<td>Creating the document not possible; source object does not exist</td>
<td>Create document not possible; source object does not exist</td>
</tr>
</tbody>
</table>

**8.2.1.5.2 Inability to Perform Action Because Object is Locked — Standard Formulations**

You use these standard formulations in cases where an action cannot be performed because the object is being used by another user.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation not possible; &lt;object&gt; locked by &lt;user&gt;</td>
<td>Operation not possible; hierarchy locked by user &amp;1</td>
<td>The operation cannot be performed because the hierarchy is locked by user &amp;1</td>
</tr>
<tr>
<td></td>
<td>Operation not possible; production order locked by user &amp;1</td>
<td>Production order is locked by user &amp;1</td>
</tr>
</tbody>
</table>

**8.2.1.5.3 Inability to Display Data — Standard Formulations**

You use this standard formulation in cases where the system cannot provide certain data to be displayed.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cannot display &lt;object&gt;; &lt;instruction or reason&gt;</td>
<td>Cannot display revision status; period already completed</td>
<td>You cannot display the revision status here</td>
</tr>
</tbody>
</table>

**8.2.1.5.4 Lack of Authorization — Standard Formulations**

You use this standard formulation in cases where a party or a user does not have the authorization to perform a certain step.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>... not authorized to ...</td>
<td>Buyer not authorized to display master data</td>
<td>No authorization to display master data</td>
</tr>
<tr>
<td></td>
<td>User &amp;1 not authorized to delete purchase order</td>
<td>The user does not have change authorization to delete the purchase order</td>
</tr>
</tbody>
</table>
8.2.1.5.5 Inappropriate Status — Standard Formulations

You use these standard formulations in cases where an object does not have the appropriate status. As a consequence, the user cannot change the object or use it in a transaction, for example.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Object&gt; not &lt;status&gt;</code></td>
<td>Business partner &amp;1 not active</td>
<td>Business partner &amp;1 has not been activated</td>
</tr>
<tr>
<td><code>&lt;Object&gt; &lt;status&gt;</code></td>
<td>Bidder &amp;1 not released</td>
<td>Release of bidder &amp;1 missing. Note: The correct status name is <strong>not released</strong>.</td>
</tr>
<tr>
<td></td>
<td>Delivery request inconsistent</td>
<td>Delivery request is not consistent Note: The correct status name is <strong>inconsistent</strong>.</td>
</tr>
</tbody>
</table>

8.2.1.5.6 Mandatory Step — Standard Formulations

You use this standard formulation in cases where a certain step must be performed.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>... required for ...</td>
<td>Mandatory step &amp;1 required for phase &amp;2</td>
<td>Perform step &amp;1 before you proceed with phase &amp;2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mandatory step &amp;1 for phase &amp;2 missing</td>
</tr>
</tbody>
</table>

8.2.1.6 Standards for Existence and Availability of Objects, Missing Data, and Search Results

8.2.1.6.1 Existence of Objects— Standard Formulations

You use this standard formulation in cases where an object already exists in the system.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Object&gt; already exists</code></td>
<td>Hierarchy &amp;1 already exists</td>
<td>There is already a hierarchy &amp;1</td>
</tr>
<tr>
<td></td>
<td>Open customer quote &amp;1 already exists for buyer &amp;2</td>
<td>An open customer quote &amp;1 exists for buyer &amp;2</td>
</tr>
</tbody>
</table>
8.2.1.6.2 Multiple Existence of Objects — Standard Formulations

You use this standard formulation in cases where an object exists in the system more than once, which means it is not unique.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Object&gt; exists more than once</code></td>
<td>Party &amp;1 exists more than once</td>
<td>Party &amp;1 is not unique</td>
</tr>
<tr>
<td></td>
<td></td>
<td>There is more than one party &amp;1</td>
</tr>
</tbody>
</table>

8.2.1.6.3 Non-Existence of Objects — Standard Formulations

You use this standard formulation in cases where an object does not exist at all in the system, which means it has never been created or it has been deleted.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Object&gt; does not exist</code></td>
<td>Number range &amp;1 does not exist</td>
<td>Number range &amp;1 not found</td>
</tr>
<tr>
<td></td>
<td>Note: Number range does not exist</td>
<td>Note: This message is only correct if it is</td>
</tr>
<tr>
<td></td>
<td>means that the number range has never</td>
<td>displayed after a user-initiated search.</td>
</tr>
<tr>
<td></td>
<td>been created.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.2.1.6.4 Availability of Objects — Standard Formulations

You use this standard formulation in cases where an object exists in the system but is not available for a certain action.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Object&gt; not available</code></td>
<td>Seller &amp;1 not available for purchasing unit &amp;2</td>
<td></td>
</tr>
<tr>
<td>Note: Seller &amp;1 not available means that the seller exists in the system, but it is not available for a certain action, for example, because it has not been assigned for that action.</td>
<td>Seller &amp;1 is not defined for purchasing unit &amp;2</td>
<td></td>
</tr>
</tbody>
</table>

8.2.1.6.5 Missing Data — Standard Formulations

You use this standard formulation in cases where a mandatory field has not been filled or a required value has not been transferred in an XML message.
8.2.1.6.6 Search Results — Standard Formulations

You use these standard formulations in cases where, during a user-initiated search, a specific object was found, could not be found, or no object was found at all.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt; found</td>
<td>&amp;1 entries found</td>
<td>&amp;1 entries determined</td>
</tr>
<tr>
<td>&lt;Object&gt; not found</td>
<td>Specified material &amp;1 not found</td>
<td>Specified material &amp;1 could not be found</td>
</tr>
<tr>
<td>No &lt;Object&gt; found</td>
<td>No documents found</td>
<td>No suitable entry was found</td>
</tr>
</tbody>
</table>

8.2.1.6.7 Determination Problem — Standard Formulations

You use this standard formulation in cases where the system cannot determine a certain object or value.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Object&gt; not determined; &lt;reason&gt;</td>
<td>Product type for product &amp;1 not determined; add product type to product</td>
<td>The system could not determine the product type for product &amp;1</td>
</tr>
<tr>
<td></td>
<td>Buying organizational unit not determined</td>
<td>Buying organizational unit not found Note: This message would only be correct if it was displayed after a user-initiated search</td>
</tr>
</tbody>
</table>

8.2.1.7 Standards for Invalid Data and Value Problems

8.2.1.7.1 Invalid Data — Standard Formulations

You use these standard formulations in cases where a certain value is not valid in the current context or if a certain value is not valid at all.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>... not valid</td>
<td>Inventory valuation level &amp;1 not valid</td>
<td>Inventory valuation level &amp;1 is invalid</td>
</tr>
<tr>
<td>... not valid for ...</td>
<td>Number &amp;1 not valid for external interval &amp;2</td>
<td>Number &amp;1 is invalid for external interval &amp;2</td>
</tr>
</tbody>
</table>
### 8.2.1.7.2 Value Problems — Standard Formulations

You use these standard formulations in cases where a certain value is not permitted or wrong in a specific context.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Value&gt;</code> not permitted</td>
<td>Unit of measure &amp;1 not permitted&lt;br&gt;Note: &amp;1 is a valid unit of measure, but it is not permitted in the context of the error message.</td>
<td>Unit of measure &amp;1 not allowed</td>
</tr>
<tr>
<td>Only one <code>&lt;object&gt;</code> permitted</td>
<td>Only one buyer permitted</td>
<td>Enter exactly one party of role buyer</td>
</tr>
<tr>
<td></td>
<td>Only one quantity permitted as valuation quantity</td>
<td>Only one quantity can be the valuation quantity</td>
</tr>
<tr>
<td><code>&lt;Value&gt;</code> greater than <code>&lt;value&gt;</code></td>
<td>Item value greater than limit &amp;1 of purchase order &amp;2&lt;br&gt;Note: This indicates that the item value is greater than allowed.</td>
<td>Item value is higher than limit &amp;1 of purchase order &amp;2</td>
</tr>
<tr>
<td></td>
<td>Purchase order net amount less than minimum order value&lt;br&gt;Note: This indicates that the item value is less than allowed</td>
<td>Purchase order net amount is lower than minimum order value</td>
</tr>
<tr>
<td><code>&lt;Value&gt;</code> less than <code>&lt;value&gt;</code></td>
<td>Invoiced quantity must not be greater than delivered quantity for item &amp;1&lt;br&gt;Note: This indicates that the invoiced quantity must be smaller than or equal to the delivered quantity.</td>
<td>Invoiced quantity must be less than or equal to the delivered quantity for item &amp;1</td>
</tr>
<tr>
<td><code>&lt;Value&gt;</code> must not be greater than <code>&lt;value&gt;</code></td>
<td>Invoiced quantity must not be greater than delivered quantity for item &amp;1&lt;br&gt;Note: This indicates that the invoiced quantity must be smaller than or equal to the delivered quantity.</td>
<td>Invoiced quantity must be less than or equal to the delivered quantity for item &amp;1</td>
</tr>
</tbody>
</table>

### 8.2.1.7.3 Assignment Problems — Standard Formulations

You use this standard formulation in cases where a user or business partner has not been assigned.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>... not assigned to ...</td>
<td>Contact person &amp;1 not assigned to bidder &amp;2</td>
<td>Assignment missing for contact person</td>
</tr>
</tbody>
</table>

### 8.2.1.7.4 Role Problems — Standard Formulations

You use this standard formulation in cases where a user or business partner does not have the appropriate role.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>... does not have ... role; &lt;instruction or reason&gt;</td>
<td>&amp;1 with ID &amp;2 does not have &amp;3 role; role must be assigned</td>
<td>Bus partner role cat &amp;3 not maintained in master data for party &amp;1 ID &amp;2</td>
</tr>
</tbody>
</table>
8.2.1.7.5 Date Problems — Standard Formulations

You use these standard formulations in cases where there are problems with a date used in a business transaction.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;... date&gt; must be before &lt;... date&gt;</td>
<td>Start date must be before end date</td>
<td>Enter a start date that is before the end date</td>
</tr>
<tr>
<td>&lt;... date&gt; must be after &lt;... date&gt;</td>
<td>Submission deadline must be after end date</td>
<td>Enter a submission deadline that is after the start date</td>
</tr>
<tr>
<td>&lt;... date&gt; must be the same as &lt;... date&gt;</td>
<td>Posting date must be the same as run date</td>
<td>Enter a posting date that is the same as the run date</td>
</tr>
<tr>
<td>&lt;... date&gt; must not be after &lt;... date&gt;</td>
<td>Source of supply date must not be after supply quota arrangement date <strong>Note:</strong> This indicates that the source of supply date must be before or the same as the supply quota arrangement date.</td>
<td>Date/Time From of SourceOfSupply is later than that of SQA</td>
</tr>
<tr>
<td>&lt;... date&gt; must not be in the future</td>
<td>Receipt date must not be in the future</td>
<td>Enter a receipt date that is not in the future</td>
</tr>
<tr>
<td>&lt;... date&gt; must not be before &lt;... date&gt;</td>
<td>Source of supply date must not be before supply quota arrangement date <strong>Note:</strong> This indicates that the source of supply date must be after or the same as the supply quota arrangement date.</td>
<td>Date/Time To of SourceOfSupply is earlier than that of SQA</td>
</tr>
<tr>
<td>&lt;... date&gt; must not be in the past</td>
<td>Start date must not be in the past</td>
<td>Enter a start date that is not in the past</td>
</tr>
<tr>
<td>&lt;... period&gt; must not start before &lt;... date&gt;</td>
<td>Binding period must not start before submission deadline <strong>Note:</strong> This indicates that the binding period must start on or after the date of the submission deadline</td>
<td>Enter a binding period that starts after the submission deadline</td>
</tr>
</tbody>
</table>

8.2.1.8 Standards for Comparison Problems

8.2.1.8.1 Missing Counterpart — Standard Formulations

You use this standard formulation in cases where the counterpart of a certain object cannot be found for a comparison.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>No ... matches ...</td>
<td>No accounting notification matches payment allocation &amp;1</td>
<td>No accounting notification coincides with payment allocation &amp;1</td>
</tr>
<tr>
<td></td>
<td>No accounting notification matches supplier invoice</td>
<td>The corresponding accounting notification does not exist for supplier invoice</td>
</tr>
</tbody>
</table>
8.2.1.8.2 Difference in a Comparison — Standard Formulations

You use these standard formulations in cases where the following applies:

- Two values in a comparison are not the same
- One value in a comparison is not the same as the other
- The type or category of an object in a comparison does not match the type or category of another object

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>... differ</td>
<td>Transaction dates differ</td>
<td>Transaction dates mismatch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transaction dates do not coincide</td>
</tr>
<tr>
<td>... differs from ...</td>
<td>Currency &amp;1 in confirmation differs from currency &amp;2 in purchase order</td>
<td>Currency &amp;1 in confirmation mismatches currency &amp;2 in purchase order</td>
</tr>
<tr>
<td></td>
<td>Price of item &amp;1 in confirmation differs from that in purchase order</td>
<td>Price difference between confirmation and purchase order in item &amp;1</td>
</tr>
<tr>
<td>... and ... do not match</td>
<td>Identified stock &amp;1 and material &amp;2 do not match</td>
<td>Identified stock &amp;1 and material &amp;2 differ</td>
</tr>
</tbody>
</table>

8.2.1.9 Standards for Object Selection

8.2.1.9.1 Selecting Objects — Standard Formulations

You use these standard formulations for error situations where the end user has to select an object.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select &lt;object&gt;</td>
<td>Select the document to be deleted</td>
<td>Choose the document to be deleted</td>
</tr>
<tr>
<td>Additional &lt;objects&gt; not selected</td>
<td>Additional status profiles not selected</td>
<td>Additional status profiles not highlighted</td>
</tr>
</tbody>
</table>

8.2.1.10 Standards for Application Log Messages

8.2.1.10.1 Standards and Guidelines for Application Log Messages

Overview

This section of the style guide and its subsections provides standards and guidelines for application log messages. Application logs belong to mass data run objects (MDROs) and support mass data runs. When a run is started, it takes place in the background. At the end of the run, the application log is displayed on the user interface (UI) to retrace the results.
Most of the message texts that are used in application logs are not really error messages. They are used to provide texts for the different sections of the application log, such as headings or selection data.

**Structure of the Application Log**

Application logs consist of three sections that group the messages posted:

1. **General**
   - This section contains a summary of what happened during the mass data run and generic log statistics, for example, the number of errors.

2. **Settings**
   - This section contains information on the parameters and settings that have been selected for the mass data run.

3. **Results**
   - This section contains detailed information about the decisions taken and about the errors and warnings that occurred during the mass data run in chronological order.
8.2.1.10.2 Number of Objects — Standards for Application Log Messages

You use these standard formulations for messages used in application logs to display the number of objects that were or were not processed correctly during an application log run.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of ...: &lt;number&gt;</td>
<td>Number of procurement planning orders changed: &amp;1</td>
<td>&amp;1 procurement planning orders have been changed</td>
</tr>
<tr>
<td>Number of ... not ...: &lt;number&gt;</td>
<td>Number of procurement planning orders not changed: &amp;1</td>
<td>&amp;1 procurement planning orders were not changed</td>
</tr>
<tr>
<td>Number of production planning orders not changed: &amp;1</td>
<td>&amp;1 production planning orders could not be changed</td>
<td></td>
</tr>
</tbody>
</table>
8.2.1.10.3  Headings — Standards for Application Log Messages

1. For messages used in application logs as headings, use title case.
2. Do not put a colon (:) semi-colon (;), or period (.) at the end of the message.

<table>
<thead>
<tr>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank Statement &amp; 1</td>
<td>Bank Statement</td>
</tr>
<tr>
<td>Note: Under this heading, messages concerning bank statement &amp; 1 are listed</td>
<td></td>
</tr>
<tr>
<td>Checking Parameters</td>
<td>Checking parameters</td>
</tr>
<tr>
<td>Note: Under this heading, all relevant checking parameters are listed</td>
<td></td>
</tr>
<tr>
<td>Selection Data</td>
<td>Selection criteria:</td>
</tr>
<tr>
<td>Summary</td>
<td>Summary Details:</td>
</tr>
</tbody>
</table>

8.2.1.10.4  Selection By — Standards for Application Log Messages

You use this standard formulation to specify the selection category that defines the elements to be included in or excluded from an application log run. You use title case to formulate the message.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Name of SelectionBy node with spaces&gt;</td>
<td>Selection by Purchase Order</td>
<td>Selection by purchase order</td>
</tr>
<tr>
<td></td>
<td>Selection by Buyer</td>
<td>Selection by BuyerParty</td>
</tr>
</tbody>
</table>

1. If the node name includes the word **party**, leave it out in the message.

8.2.1.10.5  Inclusion/Exclusion — Standards for Application Log Messages

You use these standard formulations for messages used in application logs to specify the elements excluded from or included in an application log run.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exclude all matches (with) &lt;lower boundary type code&gt;; &lt;lower boundary ID&gt;</td>
<td>Exclude all matches with &amp; 1 &amp; 2</td>
<td>Exclusion code: &amp;</td>
</tr>
</tbody>
</table>
### 8.2.1.10.6 Control Parameters— Standards for Application Log Messages

You use this standard formulation for messages used in application logs to display the settings that are used for a control parameter.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Name of parameter element&gt;: &lt;Value of parameter element&gt;</code></td>
<td>Employee time valuation period closure date: &amp;1</td>
<td>The employee time valuation period closure date is &amp;1</td>
</tr>
</tbody>
</table>

### 8.2.1.10.7 Application Log Run Start/End — Standards for Application Log Messages

You use these standard formulations for messages used in application logs to specify when an application log run has started or ended.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>... started</code></td>
<td>Adjustment run started</td>
<td>Adjustment run has been started</td>
</tr>
<tr>
<td><code>... ended</code></td>
<td>Cleanup run ended</td>
<td>Cleanup run has finished</td>
</tr>
</tbody>
</table>

### 8.2.1.10.8 Success or Failure Messages — Standards for Application Log Messages

1. For messages that describe the result of an application log run, use sentence style.
2. Do not use the words `successfully` and `unsuccessfully`.
3. Do not put a period at the end of the message text.

<table>
<thead>
<tr>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;1 new period closure dates not created</td>
<td>&amp; new period closure dates created unsuccessfully</td>
</tr>
<tr>
<td>&amp;1 of &amp;2 orders created</td>
<td>&amp;1 of &amp;2 order have been created successfully.</td>
</tr>
</tbody>
</table>
8.2.10.9 ID Messages — Standards for Application Log Messages

You use these standard formulations for messages used in application logs to display the ID of an object that has been processed during an application log run.

<table>
<thead>
<tr>
<th>Standard Formulation</th>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>... ID: &lt;ID&gt;</td>
<td>Business transaction document ID: &amp;1</td>
<td>The business transaction document ID is &amp;1</td>
</tr>
<tr>
<td></td>
<td>Customer requirement ID: &amp;1</td>
<td>Customer Requirement ID: &amp;1</td>
</tr>
</tbody>
</table>

8.2.11 General Standards for Message Long Texts

8.2.11.1 General Standards and Guidelines for Message Long Texts

Overview

This section of the Style Guide provides you with the general standards and guidelines for writing message long texts.

General

1. Add a section to the Solution Documentation for long texts that contain additional information on complex system messages. These are messages in which you cannot explain within 73 characters what the user has to do to resolve an error.
2. Name the section in the Solution Documentation Additional Information on System Messages.
3. Use the structure below for each message long text.
4. Separate the different message long texts by sub-sections.

Structure of System Message Long Texts

Keyblock: Title

1. This keyblock is mandatory. It links the long text to the short text in the system.
2. For the title, use the text of the message short text.
3. Do not use variables in the document title. If the system message contains variables, rephrase it to the document title.

<table>
<thead>
<tr>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoice may be a duplicate of another invoice</td>
<td>Invoice may be a duplicate of invoice &amp;1</td>
</tr>
</tbody>
</table>

Keyblock: What Happened?

1. This keyblock is optional. It advises the user what caused the error and what implications the error has.
2. Answer the following questions:
a. What is the cause of the error message?
b. In which context does the error occur?
c. What are the implications?

**Keyblock: What Can I Do?**
1. This keyblock is mandatory. It advises the user what they can do to resolve the error message.
2. Answer the question: What can the user do to solve the error?
3. If appropriate, use a numbered list.

**Example**

<table>
<thead>
<tr>
<th>Message Short Text/Keyblock</th>
<th>Sample Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoice may be a duplicate of invoice &amp;1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Message Long Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoice may be a duplicate of another invoice</td>
</tr>
<tr>
<td>The system was unable to post the supplier invoice or credit memo and created an exception.</td>
</tr>
<tr>
<td>Check your entry and proceed as follows:</td>
</tr>
<tr>
<td>1. Open the <strong>Supplier Invoice</strong> editor and go to the <strong>Exceptions</strong> tab. Before resolving the exception, if you require further information from another person, click <strong>Forward for Clarification</strong>.</td>
</tr>
<tr>
<td>2. Compare the invoice details with the similar invoice already posted in the system.</td>
</tr>
<tr>
<td>a. If the invoice is not a duplicate, under <strong>Exception Details</strong>, click <strong>Treat as New Invoice</strong>.</td>
</tr>
<tr>
<td>b. If the invoice is a duplicate, under <strong>Exception Details</strong>, click <strong>Treat as Duplicate Invoice</strong>.</td>
</tr>
</tbody>
</table>

### 8.2.11.2 Standards for Message Long Texts for PAF BTM Tasks

**Overview**

Process Agent Framework Business Task Management (PAF BTM) tasks are used to inform a user about a manual cleanup activity that has to be done to solve a process communication error that occurred in inbound error conflict handling.

Each PAF BTM task has a task description, which provides detailed information on a fatal error that occurred and instructions on how to solve the error.

The PAF BTM task description is structured as follows:

A fatal error occurred:

<PAF BTM Error Message Short Text>

<PAF BTM Error Message Long Text>

The message short text is used to briefly describe the fatal error that occurred. For this text (called PAF BTM error message short text in this document), the **General Standards for Message Short Texts** [page 585] and the **Standard Formulations for Message Short Texts** [page 588] apply.

The message long text provides detailed information and instructions on how to solve the error. For this text (called PAF BTM error message long text in this document), follow the standards and guidelines listed below.
General Standards

1. Include these message long texts in the section of the Solution Documentation named **Additional Information on System Messages**. Create this section if it does not already exist.
2. Formulate the message long text according to the keyblocks described below.
3. Use the SAP solution terminology.
4. If you want to use a variable, use the following format: &v1&. Do not use the variable format that is usually used for message short texts (&1).

<table>
<thead>
<tr>
<th>Example of Correct Usage</th>
<th>Example of Incorrect Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the &amp;v1&amp; project</td>
<td>Open the &amp;1 project.</td>
</tr>
</tbody>
</table>

5. Do not enter more than four variables.
6. Do not use abbreviations.

Keyblock: Context Information

1. This keyblock is optional.
2. Answer the following questions:
   a. What is the cause of the error message?
   b. In which context does the error occur?
   c. What are the implications?
3. Only use this keyblock if the message short text does not provide enough context information on the error.

   The status of the &v1& project and the &v2& task is On Hold, Stopped, or Closed. Therefore, no further postings are permitted. As a result, a time confirmation related to the &v1& project and the &v2& task could not be processed

Keyblock: Solution

General Standards

1. This keyblock is mandatory.
2. Answer the question: What can the user do to solve the error?
3. Use one of the following standard formulations:
   - If there is only one solution, use the following:
     To solve the error, proceed as follows:
   - If there is more than one solution, use the following:
     You have the following options to solve the error:
4. Describe the solution using a numbered list.
   Use the standard formulations in the sub-section below, if the message refers to one of the following:
   - Errors that refer to handling lock issues
   - Errors that occur when an inbound XML message that has been sent via a third party interface cannot be processed
5. In the second last step, describe how the system responds to the user action.

6. In the last step, describe what the user needs to do on the Process Communication Error screen to finalize the task. The user has the following options:
   - Restart process
   - Reject process
   - Synchronize process
   - Mass restart process
   - Abort process

   For more information, see the Keyblock: Result section below.

7. If there is more than one solution to solve the error, describe all of them and introduce each of them with the following standard formulation:
   Solution <number of solution>:
   Use this solution if <describe why this solution should be used> ...
**Example of Only One Solution:**
To solve the error, proceed as follows:
1. Click Open Business Document to navigate to the &v1& project.
2. Go to edit mode and navigate to the &v2& task.
3. Change the task as follows:
   - If the project status is Stopped or Closed, change it to Released.
   - If the On Hold flag is selected for the task, deselect the flag.
4. Save the changes.
5. The system sets the project status of the &v1& project and the &v2& task to Released, and On Hold is no longer selected.

**Example of More than One Solution:**
You have the following options to solve the error:

**Solution 1:**
Use this solution if the quantity sent by the service confirmation appears to be incorrect.
1. Clarify with the responsible person for the service confirmation, whether incorrect data was entered during service confirmation.
2. Ensure that the responsible person for the service confirmation corrects the data in the service confirmation.
3. The system reprocesses the data, which is now correct.

Result: The system synchronizes the documents using the updated data of the service confirmation.

**Solution 2:**
Use this solution if the stock update given for the outbound delivery is incorrect.
1. Find the corresponding outbound delivery.
2. Make adjustment confirmation on the outbound delivery.
3. The system accepts the reported quantity of the service confirmation.

Result: The system updates the stock accordingly.

1 The message short text that belongs to this long text is “Negative stock not permitted in custodian party”. This task is created if the quantity reported in a service confirmation exceeds the allocated quantity of the material in the identified stock.

**Standard Formulation for Handling Lock Issues**

For message long texts that refer to handling lock issues, use the following standard formulations:
To solve the error, proceed as follows:

1. Open the Process Communication Error screen.
2. In the Related Errors list, identify the user locking the business document.
3. Ensure that the user closes the business document.
4. The system unlocks the document.
5. Restart the process by clicking Restart Process.

Result: The system updates the business document

The standard formulation of short texts for messages referring to handling lock issues is one of the following:

- Operation not possible; <business document locked>
- <Operation> not possible, <business document> <business document ID> locked by <user name>

Standard Formulation for Inbound Third Party Communication Errors

For message long texts that refer to errors that occur when an inbound XML message that has been sent via a third party interface cannot be processed, use the following standard formulation:

Solution 1:

Use this solution if a business document that you have received via a third party interface has to be changed by the sender of the document.

1. Check the details of the error message on the Process Communication Error screen.
2. Contact the person responsible on sender side of the message to inform him or her about the issue:
   - Party: &v1&
   - Contact Person: &V2&
   - Phone Number: &v3&
   - Sender System: &v4&
3. Ask the person on sender side to perform the relevant changes (so that the message can be processed in your system) and to send the business document again.
5. Restart the process by clicking Restart Process.

Result: The system does not process the message, and the sender needs to resend a corrected message.

Solution 2:

Use this solution if you have to make changes to your system to enable the processing of a business document that you have received via a third party interface.

1. Check the details of the error message on the Process Communication Error screen.
2. Adjust the data in your system, for example, master data or business configuration data, or edit the payload, if allowed, on the Process Communication Error screen as required.
3. Restart the process by clicking Restart Process.

Result: The system updates the business documents.

For messages referring to errors that occur when an XML message that has been sent via a third party interface could not be processed, the standard formulation of the message short text is:

Error while processing inbound third party communication

Step: Result

1. This step is mandatory.
2. Describe the result of the steps performed to solve the error:
3. Depending on the user’s options, use one of the following standard formulations:
## User Option | Standard Formulation
---|---
**Restart Process** | Result: The system updates <object, including &v1&>.  
- The standard formulation of the Result step for messages referring to handling lock issues is:  
  - Result: The system updates the business document  
- For messages referring to errors that occur when an inbound XML message that has been sent via a third party interface cannot be processed, the standard formulation of the Result step is:  
  - For Solution 1:  
    - Result: The system does not process the message, and the sender needs to send a corrected message.  
    - Restart the process by clicking Restart Process.  
  - For Solution 2:  
    - Result: The system updates the business documents.  

**Reject Process** | Result: The system does not update <object, including &v1&>, and cancels the process communication.  

**Synchronize Documents** | Result: The system synchronizes the documents using the updated data of the <object, including &v1&>.  

**Mass Restart Process** | Result: The system updates <object, including &v1&> by restarting all related process communication.  

**Abort Process** | Result: The system does not update <object, including &v1&>, and the process communication is aborted.

---

Result: The system updates the &v1& project and the &v2& task with the time confirmation.

4. If there is more than one solution to solve the error, do one of the following:  
   - If the result of each is different, add a Results keyblock after each solution.  
   - If the result of all of them is always the same, only add one Results keyblock (after the last solution).

### Keyblock: Further Information

1. This keyblock is optional.
2. If there are any follow-up activities required that are not directly related to the error solution, for example, to avoid data inconsistencies, describe them here.
3. Do not describe the follow-up activities in detail.
4. Do not use a numbered list.
5. Use the following standard formulation:  
   ```
   Note: <follow-up activity> ...
   ```

**Note:** Check if it should be permitted to add further postings to the &v1& project and the &v2& task:  
- If further postings are to be permitted, no further actions are required  
- If further postings are not to be permitted, change the task data back to the previous project status or set it to On Hold.
8.3 Reference Guides

8.3.1 Reference Guide for the SAP Solution Documentation

8.3.1.1 Introduction to the SAP Solution Documentation

This reference guide provides an overview of the main reference documentation types specific to the SAP solution. It covers the following:

- **Business Configuration Documentation**  [page 613]
  - Business Adaptation Catalog
  - Basic Data
- **Work Center/Tab Documentation**  [page 615]
  - Analytics
    - Data sources: multidimensional analytical views (MDAVs), key figure structures, and key figure grids
    - Reports
    - Key performance indicators
    - Dashboards
  - Business Task Management
  - Business Background Documentation
    - Concept Documents
    - Process Documents
  - Task-Based Documentation (How-To Documents)
  - Org Functions
  - Quick Guides
  - Work Categories
- **Health Checks and Support Case Library**  [page 620]
  - Health Checks
  - Case Documents (Troubleshooting)
- **Release Information**  [page 621]
  - Business Configuration Delta Documentation
  - What’s New Documentation

8.3.1.2 Documentation Types in the SAP Solution

**Overview**

This document provides a brief overview of the main reference documentation types specific to the SAP solution.
Business Configuration Documentation

The reference documentation for business configuration covers documentation for the Business Adaptation Catalog (BAC) and the Activity List:

- **Business Adaptation Catalog**
  This reference documentation provides information about scope selection. All scope selection topics and activities to configure the system before and after go live are reflected in elements of the Business Adaptation Catalog (BAC). These elements are business areas, business packages, business topics, and business options. BAC elements require reference documentation if they have one of the following visibility options:
  - *Always*
  - *Scoping Visible*
  - *Hidden* with option type *Blueprint Only*

- **Activity List**
  This reference documentation provides information about configuring the system before and after go live. In the activity list, reference documentation covers fine-tuning activities and basic data activities.
  - **Fine-Tuning**
    These documentation types provide users with the information they require to perform activities in the Fine-Tune phase of the *Activity List*. These documents can also include information relevant to before and after the completion of a fine-tuning activity. Depending on the complexity of the activities and the activity type, there are three types of documents that can be written for these activities:
    - *Fine-Tuning Business Background*
      This documentation type describes the activity and guides the user through the activity by listing the required steps and settings. This documentation type can be used for simple activities that lead directly to a configuration screen.
    - *Configuration Guides*
      This documentation type is suited to activities that are fact sheets. Fact sheets are a collection of related links and texts for further navigation, for example to configuration screens.
    - *Basic Data*
      This documentation type provides information about activities during the Integrate and Extend phase of the *Activity List* in business configuration. These activities involve migrating or maintaining core master data as a prerequisite for other fine-tuning, migration, or testing activities. The purpose of this documentation type is to advise a user how to perform an activity and why an activity must be carried out during the configuration phase. Detailed instructions for actually carrying out the steps within the activity can come in the form of links to other existing documents.

For information on how this documentation is structured, see [Structure of Reference Documentation for Business Configuration](#) [page 613].

Work Center/Tab Documentation

The reference documentation for work centers/tabs covers documentation for the following:

- **Analytics**
  This reference documentation provides information about analytic content that is displayed on the user interface. This includes the following types of objects:
    - Data sources: multidimensional analytical views (MDAVs), key figure structures, and key figure grids. Overview of data sources documentation provides information about the multidimensional analytical view (MDAV), key figure structures, and key figure grids that make data available for reports in an area.
such as Customer Relationship Management. This type of documentation is available in the guided quick activity for creating reports and is linked in documentation related to reports.

- **Reports**
  Report documentation provides information about a compilation of data for evaluation that can be presented in the form of a table, list, text, graph, or combination of these. It is used in detailed scoping and fine-tuning, as well as by users in the application. In scoping, the Overview keyblock of the documentation is displayed. In application Reports work center views/tabs, the full report documentation is displayed.
  Overview of reports documentation provides information about the reports that are available in an area, such as Customer Relationship Management by report category. It also contains information about relevant variables in the area. The embedded reports available in the super area are also listed. Overview of Reports documentation is available in the Reports View of work centers/tabs.

- **Business Task Management**
  This reference documentation provides information that enable them to understand and process a particular business task management (BTM) task type effectively. Each task type document explains the purpose and business process context of a particular task type in the system, why a particular user receives the task, what kind of response is required, what the follow-up process steps are, and which elements can be configured.

- **Business Background Essentials**
  This reference documentation covers the concepts and processes that form the SAP solution. Therefore, a differentiation is made between the following types:
  - **Concept**
    Concepts cover a general business concept or technical concept that has a particular implementation in the SAP solution. A concept can be specific to one view or can span several views/tabs, business areas, or even application areas.
  - **Process**
    Processes cover a sequence of logically related events or steps that have a specific outcome or result in the SAP solution. A process can be specific to one particular view/tab or can span several views/tabs, business areas, or even application areas.

- **Tasks (How-To Documentation)**
  This reference documentation provides a user with the step-by-step information they require to be able to perform a task in the SAP solution. Typically, it also includes information that the user needs to know before and after the completion of a task.

- **Org Functions**
  An org function is a predefined entity in the SAP solution that denotes functional responsibility for aspects of one or more business processes. In Organizational Management (OM), the solution key user assigns org functions to org units in order to define the role of each org unit within the context of the organization’s overall business operation. The org function assignment for a given org unit influences work center proposals for users assigned to this org unit. Each org function document describes a specific org function in terms of the associated business processes, business documents, and work centers. The purpose of org function documentation is to support the OM key user who is responsible for modeling the org structure and assigning org functions to the org units.

- **View/Tab Quick Guide**
  A quick guide provides a user with a “quick” overview of a work center view/tab. It is also the central entry point for accessing more detailed information related to the view/tab. That is, for a more in-depth understanding, the user has the option of reading the documents linked from the quick guide. It provides the following key elements:
  - Summary of what the view/tab is used for
  - Business background
  - Tasks that can be performed
Work Categories

A work category (also known as responsibility) represents a type of work related to an org function. It contains rules that determine which set of work items is distributed to which org unit or employee. This is known as the process of work distribution, where a work item can be any business document or a task in Business Task Management (BTM).

The distribution of work items is usually defined by the access rights that are given to a user. However, using work categories, work items can be directly assigned to a specific org unit or employee. Each work category provides a set of parameters that can be used to define work distribution rules for the related work items.

The SAP solution provides specific work categories for organizational work distribution and employee work distribution.

Each work category document describes a specific work category in terms of the associated business process, business documents, tasks, parameters, and work distribution rules. The primary purpose of work category content is to support solution key users who are responsible for effective distribution of work in the system.

For more information on how this documentation is structured, see Structure of Reference Documentation for Work Centers [page 615].

Health Checks and Support Case Library

This reference documentation covers the following:

- Health Checks
  A health check is an incident generated by the system when functions or applications are not operating properly. The reference documentation provides information about the cause of the incident and the information required to resolve the issue.

- Support Case Library (Troubleshooting)
  This reference documentation provides information about the symptoms, cause, and resolution of system issues that are reported as incidents in the support cockpit. Each case document covers one specific issue which may in turn be relevant to one or more related incidents.

  Case documents provide a practical, searchable medium for quickly identifying solutions to system issues and answers to questions raised by customers. Their purpose is to reduce the number of reported incidents by making solutions readily available.

  Case documents address the following types of issues:
  - How-to explanations
  - Assistance with changes to business configuration or master data
  - Workarounds
  - Explanations of system behavior

For more information on how this documentation is structured, see Structure of Reference Documentation for Health Checks and Support Case Library [page 620].

Release Information

This reference documentation includes the following:

- Business Configuration Delta Documentation
  Business Configuration (BC) Delta documentation provides information on all changes that have taken place since the last release in the configuration of the SAP solution that impact customer settings.

  The changes usually refer to settings that the customer performed during scoping and to settings made from the Activity List. These scope selections and activities are reflected in elements in the business configuration catalog (BAC). The BAC elements are known to the key user as business configuration elements.
What’s New Documentation

What’s New documentation is a series of documents that, taken as a whole, describes what is new and what has been changed or deleted in a particular version of the standard SAP solution. The first document in the series describes system-wide changes. Depending on the level of change within each area and its parts, other documents may be created for specific areas (for example, Customer Relationship Management), topics (for example, Organizational Management), or views-tabs (for example, Opportunities).

For more information on how this documentation is structured, see Structure of Release Information [page 621].

8.3.1.3 Structure of Reference Documentation for Business Configuration

Overview

The reference documentation for business configuration covers Business Adaptation Catalog (BAC) documentation and Activity List documentation:

Business Adaptation Catalog Documentation

This documentation provides information about scope selection. All scope selection topics and activities to configure the system before and after go live are reflected in elements of the Business Adaptation Catalog (BAC). These elements are business areas, business packages, business topics, and business options. BAC elements require reference documentation if they have one of the following visibility options:

- Always
- Scoping Visible
- Hidden with option type Blueprint Only

Structure of BAC Reference Documentation (Excluding Business Options for Reports)

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the title of the BAC element?</td>
</tr>
<tr>
<td>Overview</td>
<td>What function does this element provide? What are the benefits of using this element?</td>
</tr>
<tr>
<td>Relevance</td>
<td>When or why should the user select this element? What are the implications of selecting or not selecting this element? Has this element been selected by constraint?</td>
</tr>
</tbody>
</table>

Structure of BAC Reference Documentation — Business Options for Reports

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the title of the report?</td>
</tr>
<tr>
<td>Overview</td>
<td>What can the user do with the report? Why should the user use the report?</td>
</tr>
</tbody>
</table>

The documentation for business options for reports matches that of the related report documentation. For more information, see Structure of Reference Documentation for Work Centers/Tabs [page 615].
Activity List Documentation

This reference documentation provides information about configuring the system before and after go live. In the activity list, reference documentation covers fine-tuning activities and basic data activities.

Fine-Tuning Documentation

These documentation types provide users with the information they require to perform activities in the Fine-Tune phase of the Activity List. These documents can also include information relevant to before and after the completion of a fine-tuning activity. Depending on the complexity of the activities and the activity type, the following types of documents can be written for these activities:

- Activity Documentation
  This documentation type describes the activity and guides the user through the activity by listing the required steps and settings. This documentation type can be used for simple activities that lead directly to a configuration screen.

Structure of Activity Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the title of the configuration activity?</td>
</tr>
<tr>
<td>Overview</td>
<td>What does this configuration activity involve (in summary)? What are the implications for the user’s business?</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>What other configuration activities have to be performed <strong>before</strong> this activity?</td>
</tr>
<tr>
<td>Steps</td>
<td>What are the steps that the user must carry out to perform the activity?</td>
</tr>
<tr>
<td>Follow-On Activities</td>
<td>What other configuration activities have to be performed <strong>after</strong> this activity?</td>
</tr>
</tbody>
</table>

- Configuration Guide Documents
  This type of document provides a starting point for completing activities contained in an activity details screen (otherwise known as a fact sheet). It is also the central entry point for accessing more detailed information related to the activity. That is, for a more in-depth understanding, the user has the option of reading the documents linked from the configuration guide.

Structure of Configuration Guide Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of the configuration activity, followed by “- Configuration Guide”? For example, “Suppliers — Configuration Guide”.</td>
</tr>
<tr>
<td>Overview</td>
<td>What are the purpose and result of this configuration activity? Where is this activity located in the system?</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>What prerequisites must be fulfilled before completing this activity? Are there any other activities which must be completed <strong>before</strong> this activity can be completed?</td>
</tr>
<tr>
<td>Business Background</td>
<td>What are the key concepts related to this activity?</td>
</tr>
<tr>
<td>Tasks</td>
<td>For each individual configuration activity:</td>
</tr>
<tr>
<td></td>
<td>• What is the title of the configuration activity?</td>
</tr>
<tr>
<td></td>
<td>• What specific action do I need to take to perform this configuration activity?</td>
</tr>
<tr>
<td></td>
<td>• In which sequence should I perform these steps? (If applicable, also explain what the result of each successive action is.)</td>
</tr>
</tbody>
</table>
Basic Data Documentation

This documentation type provides information about activities during the Integrate and Extend phase of the Activity List in business configuration. These activities involve migrating or maintaining core master data as a prerequisite for other fine-tuning, migration, or testing activities. The purpose of this documentation type is to advise a user how to perform an activity and why an activity must be carried out during the configuration phase. Detailed instructions for actually carrying out the steps within the activity can come in the form of links to other existing documents.

Structure of Basic Data Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow-Up Activities</td>
<td>What other configuration activities have to be performed after this activity, if any, and where do they take place?</td>
</tr>
</tbody>
</table>

8.3.1.4 Structure of Reference Documentation for Work Centers

Overview

The reference documentation covers documentation for the following:

- Analytics
- Business Task Management
- Business Background Essentials
- Tasks (How-To Documentation)
- Org Functions
- Work Categories

Analytics Documentation

This reference documentation provides information about analytic content that is displayed on the user interface. This includes the following types of objects:

- Data sources: multidimensional analytical views (MDAVs), key figure structures, and key figure grids. Overview of data sources documentation provides information about the multidimensional analytical view (MDAV), key figure structures, and key figure grids that make data available for reports in a particular area, such as Customer Relationship Management. This type of documentation is available in the guided quick activity for creating reports and is linked in documentation related to reports.
### Structure of Data Source Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>For which super area is the overview of data sources? For example, Overview of Data Sources for Customer Relationship Management</td>
</tr>
<tr>
<td>Overview</td>
<td>What is the purpose of the document?</td>
</tr>
<tr>
<td>Available Data Sources</td>
<td>What MDAVs are available in the super area?</td>
</tr>
<tr>
<td>Selected Data Sources</td>
<td>Do any MDAVs, key figure structures, and key figure grids require detailed documentation?</td>
</tr>
<tr>
<td>Available Key Figure Groups</td>
<td>What key figure structures and key figure grids are available in the area?</td>
</tr>
<tr>
<td>Common Characteristics and Key Figures</td>
<td>Which basic characteristics and key figures are available that are used in multiple reports?</td>
</tr>
<tr>
<td>See Also</td>
<td>Which additional documentation exists? (cross-reference)</td>
</tr>
</tbody>
</table>

- **Reports**
  Report documentation provides information about a compilation of data for evaluation that can be presented in the form of a table, list, text, graph, or combination of these. It is used in detailed scoping and fine-tuning, as well as by users in the application. In scoping in the business options for reports, the Overview keyblock of the documentation is displayed. In the Reports views, the full report documentation is displayed.

### Structure of Report Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of the report?</td>
</tr>
<tr>
<td>Overview</td>
<td>What can the user do with the report? What should the user use the report?</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>Are there any prerequisites to running the report?</td>
</tr>
<tr>
<td>Views</td>
<td>Which views are available with the report?</td>
</tr>
<tr>
<td>Features</td>
<td>Are all of the following document subheadings relevant for the report?</td>
</tr>
<tr>
<td></td>
<td>- Running the Report: Which variables are most important?</td>
</tr>
<tr>
<td></td>
<td>- Analyzing the Report: What can the user see in the (initial state of the) report? Is the data filtered? Which characteristics can the user use to filter and analyze the content of this report further? Which elements of the report provide the possibility of opening or navigating to other related reports or business object interfaces, such as Overviews (fact sheets)?</td>
</tr>
<tr>
<td>See Also</td>
<td>Which additional documentation exists? (cross-reference)</td>
</tr>
</tbody>
</table>

- **Overview of Reports**
  Overview of reports documentation provides information about the reports that are available in a particular area, such as Customer Relationship Management by report category. It also contains information about relevant variables in the area. The embedded reports available in the super area are also listed. Overview of Reports documentation is available in the Reports view.

### Structure of Overview of Reports Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>For which area is the overview of reports? For example, Overview of Reports for Customer Relationship Management</td>
</tr>
<tr>
<td>Overview</td>
<td>What is the purpose of the overview?</td>
</tr>
<tr>
<td>Reports</td>
<td>Which reports are associated with which report category?</td>
</tr>
</tbody>
</table>
### Business Task Management Documentation

This reference documentation provides information that enables users to understand and process a particular task type effectively. Each task type document explains the purpose and business process context of a particular task type in the system, why a particular user receives the task, what kind of response is required, what the follow-up process steps are, and which elements can be configured.

#### Structure of BTM Task Type Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of this task type?</td>
</tr>
<tr>
<td>Overview</td>
<td>What kind of task type is this and why was it created?</td>
</tr>
<tr>
<td>Relevance</td>
<td>Why has this task type appeared in the user’s worklist(s)? To which areas of business is it relevant?</td>
</tr>
<tr>
<td>Response</td>
<td>What are the possible actions I can take in responding to this task?</td>
</tr>
<tr>
<td></td>
<td>What is the result of each possible action, and what are the business and workflow implications?</td>
</tr>
<tr>
<td>Configuration</td>
<td>Is it possible to configure this task type?</td>
</tr>
<tr>
<td>See Also</td>
<td>Where can the user find more information? (cross-reference)</td>
</tr>
</tbody>
</table>

### Business Background Documentation

This reference documentation covers the concepts and processes that form the SAP solution. Therefore, a differentiation is made between the following types:

- **Concept Documents**
  Concept documents cover a general business concept or technical concept that has a particular implementation in the SAP solution. A concept can be specific to one or more views or can span several business areas or even application areas.

#### Structure of Concept Documents

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of the concept?</td>
</tr>
<tr>
<td>Overview</td>
<td>When and why is the concept relevant for the target group? What is the business context?</td>
</tr>
<tr>
<td></td>
<td>Is the concept country or industry-specific?</td>
</tr>
<tr>
<td></td>
<td>What does the target group need to know about the concept?</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>Are there any steps or settings (for example, business configuration activities) that need to be carried out before execution of this concept is possible?</td>
</tr>
<tr>
<td>Freestyle Subheadings</td>
<td>Which keyblock header(s) are needed to structure the document in a meaningful way?</td>
</tr>
<tr>
<td>See Also</td>
<td>Where can the user find more information? (cross-reference)</td>
</tr>
</tbody>
</table>

- **Process Documents**
Process documents cover a sequence of logically related events or steps that have a specific outcome or result in the SAP solution. A process can be specific to one or more views, or can span several business areas or even application areas.

Structure of Process Documents

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of the process?</td>
</tr>
<tr>
<td>Overview</td>
<td>When and why is the process relevant for the target group?</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>Are there any steps or settings (for example, business configuration activities) that need to be carried out before this process can be executed?</td>
</tr>
<tr>
<td>Process Flow</td>
<td>What are the steps involved in the process?</td>
</tr>
<tr>
<td>See Also</td>
<td>Where can the user find more information? (cross-reference)</td>
</tr>
</tbody>
</table>

How-To Documentation

This reference documentation provides a user with the step-by-step information they require to be able to perform a task in the system. Typically, it also includes information that the user needs to know before and after the completion of a task.

Structure of How-To Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of the task? Precede the activity name by a verb, for example Create Opportunity.</td>
</tr>
<tr>
<td>Overview</td>
<td>What are the purpose and result of this task? Why or when should the user perform this task? Does the user need any additional background information to help them understand the concept behind the task?</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>What does the user need to do before performing this task? Which basic settings need to be in place before the user can perform this task? What are the preceding processes or tasks, and where do they take place?</td>
</tr>
<tr>
<td>Steps</td>
<td>What specific action does the user need to take to perform this task? In which sequence should the user perform these steps? (If applicable, also explain what the result of each successive action is.)</td>
</tr>
<tr>
<td>Result</td>
<td>Does the completion of this task trigger further processes or tasks, or does it have an impact on other processes or tasks? If so, what are these follow-on processes or tasks?</td>
</tr>
<tr>
<td>Alternatives</td>
<td>Is there an alternative way of performing this entire task?</td>
</tr>
<tr>
<td>Example</td>
<td>Is there an example of this task?</td>
</tr>
<tr>
<td>See Also</td>
<td>Where can the user find more information related to this task? (cross-reference)</td>
</tr>
</tbody>
</table>

Org Functions

An org function is a predefined entity in the system that denotes functional responsibility for aspects of one or more business processes. In Organizational Management (OM), the key user assigns org functions to org units in order to define the role of each org unit within the context of the organization’s overall business operation. The org function assignment for a given org unit influences work center proposals for users assigned to this org unit. Each org function document describes a specific org function in terms of the associated business processes, business documents,
and work centers. The purpose of org function documentation is to support the OM key user who is responsible for modeling the org structure and assigning org functions to the org units.

Structure of Org Function Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of the org function?</td>
</tr>
<tr>
<td>Overview</td>
<td>To which functional area is the org function related? What are the associated business processes, business documents, and tasks? What are the associated work centers?</td>
</tr>
<tr>
<td>Relevance</td>
<td>From an organizational and operational point of view, when or why should the customer use this org function? What are the implications of not using it? What are the benefits? Are there any unusual use cases?</td>
</tr>
<tr>
<td>Configuration</td>
<td>What are the business configuration prerequisites for using this org function?</td>
</tr>
<tr>
<td>Constraints</td>
<td>What constraints apply to the use of this org function within the context of the org structure?</td>
</tr>
<tr>
<td>Work Distribution</td>
<td>Which work category or work categories are associated with this org function?</td>
</tr>
<tr>
<td>Example</td>
<td>Is there an example that shows how this org function can be used?</td>
</tr>
<tr>
<td>See Also</td>
<td>Where can the user find more information related to this org function? (cross-reference)</td>
</tr>
</tbody>
</table>

Quick Guide Documentation

A Quick Guide provides a user with a “quick” overview of a view. It is also the central entry point for accessing more detailed information related to the view. That is, for a more in-depth understanding, the user has the option of reading the documents linked from the Quick Guide. The Quick Guide provides the following key elements:

- Summary of what the view is used for
- Business background
- Tasks that can be performed

Structure of View Quick Guide Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of the view followed by “Quick Guide”? For example, “Opportunities Quick Guide”.</td>
</tr>
<tr>
<td>NO HEADER</td>
<td>What is the view used for in summary?</td>
</tr>
<tr>
<td>Business Background</td>
<td>What are the key concepts and processes? What are the related business scenario(s)?</td>
</tr>
<tr>
<td>Tasks</td>
<td>To which business area does this belong? What are the tasks that the user can perform?</td>
</tr>
</tbody>
</table>

Work Category Documentation

A work category (also known as responsibility) represents a type of work related to an org function. It contains rules that determine which set of work items is distributed to which org unit or employee. This is known as the process of work distribution where a work item can be any business document or a task in Business Task Management (BTM).

The distribution of work items is usually defined by the access rights that are given to a user. However, by using work categories, work items can be directly assigned to a specific org unit or employee. Each work category provides a set of parameters that can be used to define work distribution rules for the related work items.

The system provides specific work categories for organizational work distribution and employee work distribution.
Each work category document describes a specific work category in terms of the associated business process, business documents, tasks, parameters, and work distribution rules. The primary purpose of work category content is to support key users who are responsible for effective distribution of work in the system.

Structure of Work Category Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of the work category?</td>
</tr>
<tr>
<td>Overview</td>
<td>Is the work category relevant to organizational work distribution or employee work distribution? What are the associated business processes and business documents? What are the related work centers/view?</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>What are the prerequisites for using this work category? Are there any configuration (fine-tuning) activities that are relevant for this work category? Which org function must be assigned?</td>
</tr>
<tr>
<td>Work Category Parameters</td>
<td>Which parameters are used to determine the responsibility?</td>
</tr>
<tr>
<td>Work Distribution</td>
<td>How does this work category support the distribution of work in the system?</td>
</tr>
<tr>
<td>Example</td>
<td>Is there an example that shows how this work category can be used?</td>
</tr>
<tr>
<td>See Also</td>
<td>Where can the user find more information related to this work category? (cross-reference)</td>
</tr>
</tbody>
</table>

8.3.1.5 Structure of Reference Documentation for Health Checks and Support Case Library

Overview

This document provides a brief overview of the reference documentation types specific to Health Checks and the Support Case Library (troubleshooting documentation).

Health Check Documentation

A health check is an incident generated by the system when functions or applications are not operating properly. The reference documentation provides information about the cause of the incident and the information required to resolve the issue.

Structure of Health Check Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of the error/incident?</td>
</tr>
<tr>
<td>Description</td>
<td>What happened in the system/application?</td>
</tr>
<tr>
<td>Recommended Action</td>
<td>What steps need to be taken to solve the problem?</td>
</tr>
</tbody>
</table>

Support Case Library (Troubleshooting) Documentation

This reference documentation provides information about the symptoms, cause, and resolution of system issues that are reported as incidents. Each case document covers one specific issue which may in turn be relevant to one or more related incidents.
Case documents provide a practical, searchable medium for quickly identifying solutions to system issues and answers to questions raised by customers. Their purpose is to reduce the number of reported incidents by making solutions readily available.

Case documents address the following types of issues:

- How-to explanations
- Assistance with changes to business configuration or master data
- Workarounds
- Explanations of system behavior

Structure of Support Case Library Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the name of the issue?</td>
</tr>
<tr>
<td>Short Description</td>
<td>What is the issue?</td>
</tr>
<tr>
<td>Description</td>
<td>What are the symptoms of the issue?</td>
</tr>
<tr>
<td></td>
<td>What is the context of the issue?</td>
</tr>
<tr>
<td></td>
<td>What are the implications?</td>
</tr>
<tr>
<td>Reproducing the</td>
<td>Can you outline the events or actions that led to the issue?</td>
</tr>
<tr>
<td>Issue</td>
<td>How can the issue be reproduced?</td>
</tr>
<tr>
<td>Cause</td>
<td>In key user terms, what is the cause of the issue?</td>
</tr>
<tr>
<td></td>
<td>What is the technical background?</td>
</tr>
<tr>
<td>Resolution</td>
<td>What can a support consultant or key user do to address the issue?</td>
</tr>
<tr>
<td>See Also</td>
<td>Where can the user find more information? (cross-reference)</td>
</tr>
</tbody>
</table>

8.3.1.6 Structure of Release Information

Overview

This document provides a brief overview of documenting release information covering changes to the system between releases.

Business Configuration Delta Documentation

Business Configuration (BC) Delta documentation provides information on all changes that have taken place since the last release in the configuration of the system that impact customer settings.

The changes usually refer to settings that the customer performed during scoping and to settings made from the Activity List. These scope selections and activities are reflected in elements in the business adaptation catalog (BAC). The BAC elements are known as business configuration elements.

Structure of BC Delta Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>What is the BC Delta called?</td>
</tr>
<tr>
<td></td>
<td>Use the following guidelines for the title:</td>
</tr>
<tr>
<td></td>
<td>• For a new BAC element, use “New: &lt;Name of BAC Element&gt;”</td>
</tr>
<tr>
<td></td>
<td>• For a changed BAC element, use “Changed: &lt;Name of BAC Element&gt;”</td>
</tr>
<tr>
<td></td>
<td>• For a deleted BAC element. use “Deleted: &lt;Name of BAC Element&gt;”</td>
</tr>
</tbody>
</table>
What’s New Documentation

What’s New documentation is a series of documents that, taken as a whole, describes what is new and what has been changed or deleted in a particular version of the standard system. The first document in the series describes system-wide changes. Depending on the level of change within each area and its parts, other documents may be created for specific areas (for example, for Customer Relationship Management), topics (for example, Organizational Management), or views/tabs (for example, Opportunities).

Structure of What’s New Documentation

<table>
<thead>
<tr>
<th>Keyblock</th>
<th>Answers the Following Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>To which area of the solution does this information apply?</td>
</tr>
<tr>
<td>Overview</td>
<td>To which version and area of the solution does this information apply?</td>
</tr>
<tr>
<td></td>
<td>Which topics are covered?</td>
</tr>
<tr>
<td>What Is New</td>
<td>What is the new development or function?</td>
</tr>
<tr>
<td></td>
<td>What are the business benefits of using the new development or function for the customer?</td>
</tr>
<tr>
<td></td>
<td>How does the new development or function affect the user interface (for example, fields, menu paths, and screen title)? Only mention changes that are relevant for the customer.</td>
</tr>
<tr>
<td></td>
<td>What actions must the customer take to use the new function (for example, in business configuration)?</td>
</tr>
<tr>
<td></td>
<td>How does the new development or function affect existing data such as master data or transactional data?</td>
</tr>
<tr>
<td></td>
<td>What is the impact of this new development or function on other areas of the solution (for example, on business processes or scenarios)?</td>
</tr>
<tr>
<td>What Has Changed</td>
<td>What is the changed development or function?</td>
</tr>
<tr>
<td></td>
<td>What are the business benefits of using the changed development or function for the customer?</td>
</tr>
<tr>
<td></td>
<td>How does the changed development or function affect the user interface (for example, fields, menu paths, and screen title)? Only mention changes that are relevant for the customer.</td>
</tr>
<tr>
<td></td>
<td>What actions must the customer take to use the changed function (for example, in business configuration)?</td>
</tr>
<tr>
<td></td>
<td>How does the changed development or changed function affect existing data such as master data or transactional data?</td>
</tr>
<tr>
<td></td>
<td>What is the impact of this change on other areas of the solution (for example, on business processes or scenarios)?</td>
</tr>
<tr>
<td></td>
<td>Which relevant terms have been changed?</td>
</tr>
<tr>
<td>Country-Specific Changes</td>
<td>What is the new/changed development or function for a specific country?</td>
</tr>
<tr>
<td></td>
<td>What are the business benefits of using the new/changed development or function for the customer?</td>
</tr>
<tr>
<td></td>
<td>How does the new/changed development or function affect the user interface (for example, fields, menu paths, and screen title)? Only mention changes that are relevant for the customer.</td>
</tr>
<tr>
<td></td>
<td>What actions must the customer take to use the new or changed function (for example, in business configuration)?</td>
</tr>
<tr>
<td></td>
<td>How does the new development or changed function affect existing data such as master data, or transactional data?</td>
</tr>
<tr>
<td></td>
<td>What is the impact of this change on other areas of the solution (for example, on business processes or scenarios)?</td>
</tr>
<tr>
<td></td>
<td>Which relevant terms have been changed?</td>
</tr>
</tbody>
</table>

See Also: Where can I find further information about related topics (for example, related BC Delta documents)?
9 Glossary

access context
The context in which the access to business data in a work center view can be restricted, for example, company, site, sales, or employee.

access control
A method of allowing and restricting user access rights for a work center view based on the access control list of a business object. For example, instance-based access control can be used to allow access to company-specific data only.

access control list
A list of access groups that is needed to perform authorization checks for business object instances.

action
A script file that executes business logic at a particular point in time within the business object’s lifecycle. Actions can be triggered from the user interface or from events or other actions.

action code
A coded representation of an instruction to the recipient of an XML message. For example, 01 is the action code for Create, 02 for Change, and so on.

activity list
A structured collection of mandatory and critical tasks, generated from the customer’s scoping, that must be completed either before the solution can go live for the first time or before changes can be implemented later in the solution life cycle. The activity list may also include optional activities that can be completed at any stage.

add-on solution
A solution that enables the implementation of additional business logic in SAP’s on-demand solution and the development of enhanced functions for business users.

Administration toolbar
A set of functions that allows administrators to manage solutions.
administrator
A system user that has administrator rights in SAP’s on-demand solution. You use this user in the SDK if you want to define analytics content or create a mashup. To use the key user, you enable key user mode for a solution in the SDK.

advanced search
A search that allows the user to combine free text search and attribute search. The user can specify the search, display, and filter options to locate the item they require.

alert
A task with the highest available priority that requires immediate user action.

alternative key
An annotation with which you define a unique key of a business object node. You can it in an association and it is stored as an additional key field in data searches and queries.

Analytics
Analytics provides you with the necessary infrastructure for the analysis of your enterprise. A set of predefined reports from all business areas is available that cover the most important reporting needs.

anchor
A stable point to which any user interface modifications to SAP floorplans are attached.

Application and User Management
A work center that allows key users to perform application administration tasks and to manage access rights.

application-to-application communication
A process by which messages containing information about updates to business documents are sent between process components within the system. For example, when creating a sales order, the system updates related business documents within the system.

approval notification
An electronic message sent to the requester to indicate that approval has been granted or rejected by the authorized approver.
approval process
One or multiple steps by which a business document is addressed to one or more employees responsible for approval, including any defined conditions.

Approval Processes
A work center view in which you can display, edit, copy, and delete versions of approval processes related to business documents.

approval status
An indicator of the step reached in the approval workflow of an object, for example, an invoice that has to be reviewed by an authorized approver could be listed as "waiting for approval".

approval status attribute
An element in a business object that is used to store the business object’s approval status.

approval task
A part of an approval process. Approval tasks are approval, approval/rejection notifications, and revision.

Approval Wizard
A guided activity for creating a default approval process for a custom business object in the SDK/studio.

approver
An employee responsible who has been assigned to authorize or endorse an object, such as an invoice or a leave request, at a given step in the approval process.

assigned work centers
A work center which a user can access to do their work.

association
A relationship between two business object nodes.

asynchronous communication
A form of electronic communication that does not take place in real time. E-mail is an example of electronic asynchronous communication.
availability

The capacity of a service or a component to fulfill its required function at a specific moment or over a set time period. Usually, this is measured using an availability ratio, which compares the length of time that the service has been available with the duration agreed with the customer.

BAC element

A project item for business configuration such as a business option or a business topic that is required to activate a solution in a customer system. In the standard SAP solution, a BAC element is a scoping element and can refer to any level of the business adaptation catalog hierarchy, including business packages, business topics, and business options.

BC-Set Wizard

A guided activity for creating a business configuration set based on business configuration objects.

browser-based report

A type of report that can be viewed in a Web browser.

BTM Wizard

A guided activity for creating an approval task.

build

The software version of the SAP Business ByDesign studio.

business adaptation catalog

A central structure that uses business language to describe and organize all the capabilities of SAP’s on-demand solutions. It structures the capabilities into a hierarchy of business areas, packages, topics, and options. Customers and prospects make selections from the catalog based on their specific business needs. The catalog enables change both during the initial setup of the solution and throughout its life cycle.

Business Analytics

A work center that enables key users to create and assign reports, and to create and edit key performance indicator evaluations.
**business area**
The first and highest level in the hierarchy of the business adaptation catalog. It represents the collective group of functions that the solution supports in a specific part of the company. Examples are sales and human resources. Each business area contains a number of business packages.

**business configuration**
The process of adapting the solution to the day-to-day business requirements of the customer at any time in the life cycle of the solution without the need for technical skills.

**Business Configuration**
A work center where users can configure the solution to meet their specific business requirements. You can access the Business Configuration work center both before and after initial go live, but different functionality is available to you depending on what stage you are at in configuring your solution.

**business configuration content**
Project items that are anchored in the business adaptation catalog to allow key users to implement a solution in the production system. Examples include business options and business topics, business configuration sets, and business configuration views.

**business configuration object**
A project item for business configuration that represents a configuration table and is used to create business configuration sets. A business configuration object is a business object category.

**Business Configuration Object Wizard**
A guided activity for creating a business configuration object that is then used to create a business configuration set.

**business configuration set**
A technical container for the configuration settings and corresponding values that are associated with a business option. The content in a BC set is deployed to the runtime environment where it is loaded into the underlying configuration tables. Each configuration setting in a BC set has attributes that specify whether it is hidden, visible, or changeable for customers.

**Business Configuration Set Wizard**
A guided activity for creating a business configuration set based on a business configuration object.
**Business Configuration Wizard**

A guided activity for creating business options and business topics and anchoring this business configuration content in the business adaptation catalog.

**business object**

A model of a real-world object, for example, an employee or a sales order in business application systems.

**business object extension**

An SAP business object that has been extended to include additional fields.

**business object instance**

A representation of the object type of a business object. For example, the sales order numbers 12345 and 56789 are instances of the Sales Order business object type.

**business option**

The fourth level in the hierarchy of the business adaptation catalog. Business options can be hidden or can appear in scoping, fine-tuning, and/or the solution proposal. It represents the most detailed decision level involved in defining or adapting a solution capability. Each business option specifies a particular way to perform a function. It is based on predefined content. Examples are General Ledger Accounting and Customers. Each business option belongs to a business topic.

**business package**

The second level in the hierarchy of the business adaptation catalog. It covers the organization, processes, management support and master data of a given application area. Each business package belongs to a business area and contains a number of business topics.

**Business Task Management**

An integral part of the system which manages the prioritization and assignment of tasks to the appropriate users.

**Business Task Management**

A work center view that enables users to monitor unassigned items and automated tasks with errors.

**business topic**

The third level in the hierarchy of the business adaptation catalog. It relates to specific functions within an application area. Examples are Account Management and Time Administration. Each business topic belongs to a business package and contains a number of business options.
**business user**
A user that can log on to the system. You grant access rights to users by assigning work centers or work center views to their business user.

**business-to-business**
Electronic transactions between one business and another, for example, between a supplier and a retailer. Business-to-business transactions are typically high volume and involve a long-standing business relationship.

**calculated key figure**
A key figure that is determined using calculation rules or formulas.

**change transaction**
An action that allows you to modify an SAP floorplan. For example, you can use a change transaction to add a view to a standard work center.

**characteristic**
A field according to which values are selected. Characteristics are alphanumeric, numeric, or text values. Examples include Product ID, Supplier, and Purchase Order Status.

**chart of accounts**
A systematic listing of all general ledger accounts used by an organization of one or more companies, normally in alphanumeric order.

**clarification request**
A manually created task used for asking another user for more information on an item.

**code list**
A field with a set of predefined values that is of data type Code.

**Code List Data Type Wizard**
A guided activity for creating a static code list in the SDK/studio.

**collection**
A table that is defined on the basis of business object nodes or node elements, for example, a collection of sales order items.
**combined data source**
A type of data source that combines two or more data sources. The data sources of which the combined data source consists usually have a set of common characteristics but semantically different key figures.

**common task**
A task that you perform frequently, such as creating a sales order.

**Communication Scenario Definition Wizard**
A guided activity for defining a communication scenario in the SDK/ studio.

**compliance**
The process of conforming to a specification, standard or law, for example a law relating to data protection.

**composition**
A navigational relationship from a superordinate to a subordinate node of a business object.

**condition**
A predefined filter to restrict the data displayed to a specified characteristic and key figure. There are two types of conditions: simple and ranked.

**Configuration Explorer**
A tool window in the UI designer that displays the SAP repository content.

**content area**
The area of the application window for performing user tasks. The content area is visible on every page and changes when the user navigates from one page to another.

**continuous improvement**
A commitment by SAP to regularly analyze all aspects of a customer system configuration and then identify improvement potential and recommend actions. The analysis covers a range of issues including KPIs and bottlenecks. Actions SAP can recommend include changing system settings and installing delivery patches.

**custom business configuration object**
A business configuration object created as part of a solution in the software development kit.
custom business object
A business object defined as part of a solution in the software development kit.

custom control
A user interface control created in Microsoft Visual Studio using Microsoft Silverlight. The control can be used in the software development kit to enhance screens of SAP’s on-demand solution.

custom pane
A user interface pane created in Microsoft Visual Studio using Microsoft Silverlight. The control can be used in the software development kit to enhance screens of SAP’s on-demand solution.

custom query
A query created as part of a solution in the software development kit.

custom reuse library
A collection of user-defined functions. You can use a library to organize the code in your solution or to reuse code in several script files.

customer-specific solution
A business solution that is designed and developed by an SAP partner to enhance SAP’s standard on-demand solution for a specific customer.

data source
An object containing key figures and characteristics, which provides a multidimensional view of business data for reporting.

Data Source Wizard
A guided activity for creating and defining a data source. In addition, you can define transformation rules, define aggregation behavior for key figures, and reference other data sources.

default approval process
A process created in the SDK for a custom business object for use in SAP’s on-demand solution. A default approval process has one approval step, no conditions, and unlimited validity.
dependent object
A reusable part of a business object that cannot stand alone but can only be used in the context of the business object.

deploy a solution
Upload a solution to a customer system or to a test system.

deploy business configuration
Make business configuration content created in the studio available in the system.

deployment unit
A piece of software that can be operated on a separate system isolated from other pieces of software. All deployment units can communicate directly with the Foundation deployment unit.

derived name
The default naming convention provided by the software development kit to name a message data type in a Web service operation. A derived name is based on the names of the service operation and the business object element on which the service operation is allowed to be performed.

Design Data Sources
A work center view in which you can create, edit, and delete data sources and create reports.

Design Reports
A work center view in which you can create, edit, delete, and assign reports to work center views.

details area
An area of the screen providing details about a user selection. For example, if a user selects a row in a work list, the details area provides details about that row.

developer desktop
The user interface of the SAP Business ByDesign studio that presents and provides access to development tools in an integrated development environment (IDE).
**dialog box**
A box on the screen that a user must respond to before resuming work. Sometimes the user prompts the dialog box to display and sometimes it displays without prompting.

**dimension**
A criterion by which characteristics are categorized.

**dropdown list**
A list of entries that a user can display by clicking a down-facing arrow to the right of the first entry in the list.

**element**
A component of the business adaptation catalog. An element is a general term that can refer to any level of the catalog hierarchy, for example, a business package, topic, or option.

**element**
Part of a business object. Elements are displayed as fields on the UI.

**embedded component**
A reuseable user interface component that can be added to other floorplans.

**embedded report**
A report that is embedded in a specific context, such as in an overview document.

**Employee Work Distribution**
A work center view that enables the key user to define users responsible for specific work items.

**endpoint**
A location for accessing a Web service using a specific protocol and data format.

**enhancement implementation**
An extension of an enhancement option that allows customers to apply business rules that meet specific requirements such as country-specific requirements.
**enhancement option**

An entity provided by SAP to allow the standard behavior of a specific application, such as Financials, to be enhanced without modifying the standard solution.

**enhancement option type**

An attribute that indicates whether an enhancement option is for single use or multiple use.

**error message**

A message informing a user about an error. The message typically explains what went wrong, why it went wrong, and what the user can do about it. Most error messages display in the message area of the screen. However, if the system requires that the user must respond to an error, the error message displays in a dialog box.

**event**

A script file that executes business logic at a particular point in time within the business object’s lifecycle, for example, before save or after modify.

**exception**

A deviation from a defined threshold value or interval. Exceptions can be applied to data, results, or a combination thereof and can be displayed as an arrow, background color, or traffic light.

**exception aggregation**

Aggregation behavior, which deviates from standard aggregation (such as summation), for values of a calculated key figure. The deviation is only valid for a specified characteristic.

**Extensibility Explorer**

A tool window that displays the anchors attached to a floorplan.

**extension field**

A field that has been added to an SAP business object. Extension fields can be used, for example, in screens, forms, reports, and search categories.

**extension scenario**

A representation of all business contexts that are part of a process flow in which an extension field may be involved.
**External Web Service Integration Wizard**
A guided activity for integrating an external web service in the SDK/ studio.

**favorites**
A location where hyperlinks to commonly viewed content (for example, reports and business documents) or Web pages can be stored and easily accessed.

**file input**
A general method for uploading external data files on specific interfaces into an SAP Business ByDesign system.

**file input run**
A mass data run that uploads data files from external applications into an SAP Business ByDesign system, using existing service interfaces.

**filter**
To retrieve data that matches user-defined filter criteria.

**filter**
A project item that belongs to an enhancement implementation that allows users to work with a specific set of data.

**fine-tuning**
The configuration of the solution to meet customer requirements that exceed the level of predefined content in the software. It is based on input and selections made during scoping and occurs before the solution goes live.

**fine-tuning project template**
A project item for business configuration that contains predefined fine-tuning activities for a specific scenario that is defined by SAP, such as Financials: Chart of Account.

**first implementation project template**
A project item for business configuration that contains predefined scoping elements and fine-tuning activities. Customers can apply this template to define the initial scope of their solution when they implement SAP Business ByDesign for the first time.

**flag**
The marking of a business document for later reference, using a flag icon.
**foreign key relationship**
A relationship between two business object nodes for which the alternative key of the target node is specified.

**form data type**
A form-specific data type based on an existing data type.

**form data type extension**
An SAP form data type that has been extended to include additional fields.

**Form Data Type Wizard**
A guided activity for creating a form data type extension.

**Form Template Selection**
A work center view that enables key users to create form template rules for business documents.

**front-end script**
A programming language used in SAP Business ByDesign User Interface Designer to define rules and logic that determine the behavior of user interface (UI) controls. For example, you can write a front-end script that enables an Edit button only when a user selects a field on the UI.

**Gallery**
A work center view that allows you to browse reports.

**general ledger account**
A record of a company’s financial transactions used for reporting in accordance with a set of books.

**global solution**
A solution that can be transported and therefore can be deployed to a customer.

**guided activity**
A secondary window consisting of a series of interactive screens that guides you through each step of a particular activity. For example, the New Invoice guided activity guides the user through the steps involved in creating a new invoice.
**HTML mashup**
A mashup that integrates HyperText Markup Language (HTML) or JavaScript content as embedded content. Examples include an online map, a widget, or social networking information.

**icon**
A small pictorial or graphical image that represents an object, action, or status.

**implementation project template**
A project item for business configuration that contains predefined scoping elements and fine-tuning activities.

**incident trace**
A trace that is attached to a support request.

**Incidents**
A work center view in which a key user resolves incidents or forwards them to the provider for resolution.

**information icon**
A symbol on the user interface that can be found next to entry fields. When you click on the icon, the system displays useful text.

**inport**
A port configured to receive a navigation key from the source user interface. This key is used by the target user interface to fill business data.

**integrated solution**
A solution that enables the integration of applications with SAP’s on-demand solution. An example is an external Web shop application that creates sales orders in SAP’s on-demand solution.

**IP address**
A unique address used to identify a node in an IP network.

**item**
A business object, record, or piece of information in the system, for example, Account, Contact, Lead, or feed update.
joined data source
A type of data source that joins two or more data sources. The joined data source consists of a subset of characteristics that are the same in the data sources to be joined along with semantically different key figures.

key figure
A field according to which values are selected. Key figures are numeric values that have a unit of measure or currency assigned. Examples include Invoice Net Value and Purchase Order Quantity.

key figure grid
A group consisting of a key figure structure and a characteristic structure for reporting on two axes.

key figure group
A key figure structure or key figure grid.

key figure structure
A group of predefined key figures for reporting on one axis.

key user
A system user that has administrator rights in SAP Business ByDesign. You use this user in the SAP Business ByDesign studio if you want to define analytics content or create a mashup. To use the key user, you enable key user mode for a solution in the studio.

lightweight solution
A solution that enables the adaptation of forms, report, and screens used in SAP’s on-demand solution.

local trace
A trace that is run in the system in which it was created.

localization
The process of adapting a product to meet the needs and expectations of a specific country or region, with respect to language, laws, business practices, or culture.

log out
To end a previously authenticated session of the application. When you log out, you can no longer use the application. To use the application again, you must log in again.
**maintenance mode**
The status of a solution after it has been assembled. You can make changes to the solution in a patch but only restricted changes to certain content types. With these change and delete restrictions, SAP wants to avoid loss of data as well as inconsistencies and errors in the productive customer’s solution.

**mashup**
A Web service or application that has been integrated into an SAP on-demand solution. Mashups combine and transform internal business data with information or services provided by external online service providers.

**Mashup Authoring**
A work center view in which key users or administrators can create and edit mashups.

**mashup in-port**
The incoming data connection between a screen and a mashup. The mashup in-port is used to input business data from the system into the mashup. This data comes from the screen out-port and can contain multiple parameters.

**mashup out-port**
The outgoing data connection from a mashup to a screen. The mashup out-port is used to pass the result of the mashup to the system. This is then mapped to the screen in-port.

**mashup solution**
A solution that enables the integration of Web-based applications or services into the user interface of SAP’s on-demand solution. An example is the integration of a Web application for maps and route planning into a user interface displaying the address of a business contact.

**mashup type**
The categorization of mashups by the technology on which they are based. Examples include a URL mashup or HTML mashup.

**mashup Web service**
A third-party Web service that a key user or an administrator can integrate into an SAP on-demand solution so that it can be used by mashups.

**Mashup Web Services**
A work center view in which key users or administrators can integrate third-party Web services into an SAP solution so that they can be used by mashups.
mass data run
The automatic mass processing of a business transaction. An invoice run is an example of a mass data run.

mass data run object
A business object specifying an automatic mass processing of a business transaction. It contains the selection criteria and the processing results.

mass-enabled script file
An action or event that can be executed for multiple instances of a business object node.

message definition
A system notification and its attributes defined to assist customers who work with an enhancement implementation.

method tooltip
An on-screen explanation of a method and its parameter definition.

micro-vertical solution
A solution that consists of an add-on solution or an integrated solution, or a combination of both, designed for a particular industry or business area.

modal dialog
A small window that helps the user to perform a task. It is triggered from a calling application and has to be completed before the user can continue to work in the calling application.

multiple-use enhancement option
Identifies an enhancement option that can be implemented more than once in the same solution.

My Solutions
A tool window in the SDK in which you can create and access solutions in the repository of SAP’s on-demand solution.

navigation
The means by which a user can move around, within and between, screens. Effective navigation gives the user as much freedom as possible to switch from one screen to another. Navigation elements include menus, hyperlinks, and standard buttons such as Back, Forward, and Exit. Note: The browser Back and Forward functions are not supported in the current solution scope.
notification
Defined as part of the default approval process. Notifications are sent to the participants of an approval process to inform them of a task to be carried out. In the SDK, you can define an approval notification and a rejection notification.

Notification Rules Wizard
A guided activity for enabling notification rules for a custom business object in the studio.

object instance floorplan
A combination of UI patterns arranged in a specific order that allows users to create, delete, view, and edit attributes and associations of a business object.

object work list
A list pane that is embedded in a work center view.

operand
A numeric value upon which an operation is performed.

operator
A symbol that describes an operation to take place between two or more values.

outport
A port configured to pass the navigation key from the source user interface to the target user interface.

Output Channel Selection
A work center view that enables key users to create output channel rules for the process-integrated output of business documents.

parameter binding
Mapping of a URL parameter to a screen out-port.

password
A combination of alphanumeric characters used by a user to gain access to the system.
patch
Changes or corrections to a customer-specific solution after it has been assembled.

permissions
A set of policies describing whether an identity is allowed to access a certain program or piece of information. The permissions, also called access rights, are grouped into work centers. The work centers can then be assigned to identities, for example, to a user.

personalize
To change or customize something to suit your needs. For example, change the way that your desktop is configured so that you have easy access to the applications you work with.

placeholder
A symbol or chain of symbols that, at runtime, is replaced by text or numbers.

port binding
A grouping of predefined screen out-ports and in-ports that can be used to create a mashup. The port binding defines the type of information that can be used in the mashup and defines on which screens you can use the mashup.

port type package
A blueprint for an inport or an outport. A port type package hosts one or more port types.

process communication error
An error that occurs if the system cannot process a message sent from one business object to another during an update of business data.

Process Communication Errors
A work center view that allows key users to monitor application-to-application and business-to-business communication errors.

process extension scenario
A type of extension scenario that links the data from one business context to other related business contexts. You create an extension scenario from a list of predefined extension scenarios.
**production environment**
A reserved space that is used by an individual prospect or customer to store all information needed to operate its live solution and run its business. Since any changes that are made here have a direct affect on the running system, the customer typically performs changes and any related testing activities in a separate environment before doing so in the live production environment.

**production system**
A system used by the customer to work with the solution after it has gone live. This system contains all of the elements required to run the customer’s business processes using SAP Business ByDesign and can be adapted in the Business Configuration work center using immediate changes and change projects.

**production tenant**
The reserved space where a company stores all information needed to operate its live solution and run its business. Each company has its own unique production tenant, which only they can access.

**project**
A container that is used to organize all items of a solution, for example, business objects, screens, and business configuration content.

**project item**
A file in a project, for example, a business object, a screen, or a form.

**provider**
The party responsible for providing IT services or a product to a customer. This can be performed by an in-house provider or a third party.

**public solution model**
Contains all entities in SAP’s on-demand solutions that are released for external consumers. External consumers can be partners who develop solutions on top of SAP’s on-demand solutions or administrators who create new reports.

**Query Wizard**
A guided activity for creating a query, selecting query fields, and defining query parameters.

**quick activity**
A secondary window consisting of a single interactive screen that allows you to complete a particular activity.
For example, the Create Incident quick activity allows you to request support from a key user.

**quick create floorplan**
A preview of detailed information on a thing that a user can see without navigating away from the UI currently displayed.
Data shown in a quick create view cannot be edited.

**quick view floorplan**
A slide-in window, which allows users to edit data without navigating away from the UI currently displayed.

**refresh**
To update a screen, or part of a screen, so that it displays the most current information.

**report**
A compilation of data for evaluation where online analysis and drill-down is supported in every possible direction and on a very detailed level. Analytical reports are ideal for detailed root cause analysis.

**report category**
A category by which reports are grouped from a business perspective.

**report incident**
A function that allows the user to report an incident from any screen within the application.

**report wizard**
A guided activity with which you can create or change your own reports.

**Repository Explorer**
A tool window that displays SAP content, such as business objects and data types, which is released with the public solution model (PSM). It also displays the documentation for the different entities.

**Repository View**
A tool window in the SAP Business ByDesign studio in which you can create and access solutions and view SAP content in the SAP Business ByDesign repository.

**requester**
A person who creates an approval process in SAP’s on-demand solution.
reset
A function that enables you to return to the original state of a document or application.

resolution
The action taken to solve an incident. Examples of resolutions include patches and work-arounds.

REST Web service
An HTTP-based Web service that is based on REST (Representational state transfer) architectural concepts.

retention time
Period of time data are stored in a system.

reuse library
A collection of functions provided by SAP that you can use to perform common tasks such as date conversion or time calculation.

reuse library function
A self-contained module of code that operates on values passed to it and typically returns a result.

revision
Task that is part of the approval process. It is sent to the person who created the approval process and informs him or her that the approver has requested changes to the request.

revision task
The task created by the system requiring a user to change data entered because it contains errors or because a change is necessary.

rollover explanation
A brief text that provides instructions or other explanatory information about UI elements, such as fields, buttons, column headers, tab titles, radio buttons, and check boxes, in cases where the label is not self-explanatory. A rollover explanation is displayed when the user points the mouse cursor on the UI.

RSS
A method for publishing content such as news headlines, blogs, or podcasts on the Web using an XML (Extensible Markup Language) file format.
The content is automatically updated on a regular basis, and alerts users to new information.

**RSS Feed**
A document that is either the entire text or the summary of the content of an associated Web site.

**RSS/Atom Web service**
A Web feed using Really Simple Syndication (RSS) or Atom standards.

**sandbox solution**
A solution that is used for testing purposes. It cannot be transported and therefore cannot be deployed to a customer.

**SAP Business ByDesign data type scripting language**
A programming language used in the SAP Business ByDesign studio to define form data type extensions.

**SAP Business ByDesign repository**
A central location in which SAP entities and partner solutions are stored.

**SAP Business ByDesign scripting language**
The programming language used in the SAP Business ByDesign studio to define business objects and business object extensions and to implement the business logic.

**SAP Business ByDesign studio**
A software development kit (SDK) that enables SAP partners to adapt and enhance the solution capabilities of SAP Business ByDesign.
The development tools provided are presented in an integrated development environment (IDE) based on Microsoft Visual Studio.

**SAP Business ByDesign User Interface Designer**
A tool used to design or enhance the user interface of the SAP Business ByDesign application.

**SAP reuse library**
A collection of functions provided by SAP that you can use to perform common tasks such as date conversion or time calculation.
scalable solution
A business solution that is designed and developed by an SAP partner to enhance SAP’s standard on-demand solution. Customers of SAP’s on-demand solution can buy scalable solutions through the SAP Store.

scoping
The process of matching business requirements to the capabilities of the solution, using the business adaptation catalog.

scoping question
An important question the customer answers during scoping to select or deselect a business option from the business adaptation catalog. An example of a scoping question for the Data Migration business topic is “Do you want to migrate data using the built-in migration tool?”

screen out-port
The outgoing data connection from a screen to a mashup. This allows the data from the screen to be passed to a mashup in-port.

script file
A project item that contains business logic for a solution.

scripting language
The programming language used in the SAP Solutions OnDemand Studio to define business objects and business object extensions and to implement the business logic.

search
To look for specific data in business documents, worklists, or the system.

search category
A group of objects with a similar theme that allows users to limit the scope of a search. Categories include Files, Reports, Business Objects, and People, for example employees.

search criteria
The specific values that you look for while doing a search. For example, if you search for all documents created on a certain date, that date is the search criteria.
search result
A list generated after the user specifies the search criteria and executes a search.

section group
An area on the UI that contains information such as fields, links, or lists that belong together semantically.

segregation of duties
A measure used in the prevention of fraud by separating business processes so that more than one person is responsible for completing a task. This is done by assigning access rights and distributing responsibility between several users in the aim of reducing errors and preventing fraud.

service integration
Synchronous or asynchronous message-based communication between a business object and another communication partner, such as a different system, by means of Web-service technology.

service integration definition
A project item containing service integration settings.

Service Integration Wizard
A guided activity for creating message-based communication between a business object and another communication partner, such as a business object in a different deployment unit or a different system.

service level agreement
An agreement with the customer for the assurance of a specific service within a predefined period of time. Service Level Agreements (SLA) define the attributes for service product (for example, maintenance, hotline) that have been agreed upon with the customer in service contracts. The SLA confirms different parameters, such as response time, availability time and system availability. The SLA effects the pricing for each contract item and the date calculation in the service order, to which the relevant contract items refer.

Session Administrator
A tool window that displays active user sessions and provides functions for deleting sessions and item locks.

severity
The scale of importance attached to an incident. Three different levels of severity can be associated with an incident: low, medium and high.
severity level
An indication of the severity of an incident to ensure it is prioritized correctly.

shortcut
Quick access to a screen without using the common navigation path.

single-use enhancement option
Identifies an enhancement option that can only be implemented once in the same solution.

SOAP Web service
A Simple Object Access Protocol (SOAP)-based Web service.

solution
A container that is used to organize all items of a solution, for example, business objects, screens, and business configuration content.

Solution Explorer
A tool window in which you can add, edit, and delete project items.

solution proposal
A document attached to an incident that describes a possible solution to the issue. The processor attaches one or more solution proposals to an incident to provide the requester with information about how to solve the issue.

solution template
A container that is used to organize items that can be reused in customer-specific solutions.

solution variant
A subset of functions developed in a solution that customers can buy as individual solutions through the SAP Store online marketplace.

sort
To rearrange the content of a column in a table in ascending or descending order. The rest of the content in the table is rearranged accordingly.
**source language**
The language from which text is translated. The source language supported by the SAP Business ByDesign studio is English.

**step**
A stage in a guided activity towards the completion of a task.

**structure element**
A characteristic, formula, or restriction in a characteristic structure.

**Subject**
A short description of an item, such as a task or notification.

**subview**
A view that a user can open within a view.

**system message**
A short text that provides information about system feedback to the user. System message texts are displayed in the message region as error messages, warnings, success, or information messages.

**tab**
A label that the user clicks to access content. Tabs display horizontally in a tab strip and are positioned above the content that they describe.

**tab strip**
A horizontal row of tabs.

**tag**
A non-hierarchical keyword or term that is attached to an item. Tags are used to categorize or index items.

**target language**
The language to which text is translated. The target languages supported by the SAP Business ByDesign studio are German (DE), Spanish (ES), French (FR), Portuguese (PT), and Chinese (ZH).
task
A unit of work that has to be performed by a user. Examples include Business Task Management tasks, IT Service and Application Management tasks, and Activity tasks.

task description
Text that provides a brief explanation about a task.

task type
A predefined content type in the system. It has a set of properties such as task category and related business document type which characterize the task. Each task type is related to one task category.

technical user
A back-end process or service that is used to automate technical tasks in the system.

test tenant
A reserved space that is used within the solution by a single prospect or customer for evaluation, testing, or training purposes.
It provides a safe environment, separate from the production tenant. Any changes made here do not affect the live system.

thing
The fundamental component of the current UI design concept and the conceptual entity with which end users work. Things are smart and portable work objects such as a Material or a Business Partner. They embed attributes (data or images), related objects, analytical data, and actions generated from metadata.

thing inspector floorplan
Combination of UI patterns that allows the user to get a complete view of a thing.
It shows all aspects of a thing and is a compact summary of all primary attributes, the tags, and the actions that can be carried out.

thing type floorplan
Combination of UI patterns that describes the general UI properties of a thing, such as its name and various icons on the UI, its binding against a business object, and its key.

thing-based navigation
A type of user interface interaction based on business objects defined as "things".
This type of navigation is used in SAP’s on-demand solution (except for SAP Business ByDesign).

**tooltip**
A very short text that displays the name of a UI element when the label of the UI element cannot be seen or is only partly displayed to the end user.
A tooltip is displayed when the user points the mouse cursor on the UI element.

**trace**
A trace that is attached to an incident. Incident traces are tied to the life span of the incident.

**Trace Explorer**
A tool for gathering and analyzing diagnostic information.

**trace sequence**
The series of recorded method calls and evaluated trace messages that are created as a result of running a trace.

**tracing**
The process of creating and storing detailed information about runtime events in the system.

**translatable text string**
An element of a computer readable file that represents a text, such as a field label, a field value, or another user interface text element. A translatable text string can be extracted from a file for translation into a target language.

**translated text string**
An element of a computer readable file that represents a text, such as a field label, a field value, or another user interface text element, and that has been translated into a target language.

**troubleshooting**
A set of documents that provide solutions for typical problems or user errors.
These documents are available in the advanced search and in the search for incident reporting.

**UI switch**
A control in SAP’s on-demand solution that is used to restrict the visibility of UI changes.

**unicode**
A 16-bit character set that represents commonly used characters, for example letters and digits, in digital form.
Unicode has a distinct advantage over the 8-bit character set ASCII, in that it can render a much larger set of characters. For example, Unicode can represent over 30,000 distinct coded characters, whereas ASCII can only represent 128.

**URL mashup**
A mashup that sends data from the SAP Business ByDesign system to the Uniform Resource Locator (URL) of an online service provider. The service provider uses the data, for example, to perform a search, and the results are displayed in a new browser window.

**user**
A business object created for an employee or a service again who needs to log onto the system. It contains technical information such as a password and access rights. Internal components also have user IDs assigned to them.

**User and Access Management**
A work center view that is used to edit and review user attributes and access rights. The User and Access Management view is a superordinate view of the Business Users view and Support and Technical Users view.

**user ID**
A unique identifier for an individual who interacts with the services supplied by a system.

**user interface dependency**
An action that governs the assignment of views to a user.

**user interface designer**
A tool used to design or enhance the user interface of the solution.

**user management**
The process of managing all users on the system. This involves applying basic settings such as user names and passwords, assigning users to work centers, and placing restrictions on user access.

**validation**
A script file that returns either true or false based on a specified condition. Validations are triggered on the execution of events or actions. A validation can raise a message or stop further processing of an event or action.
value selection

The value selected to which a characteristic is restricted. The data shown in a report is restricted to the values selected so that not all data is displayed.

variable

A parameter of a report that restricts a characteristic to one or more specified value selections.

Web service

A Web-based application programming interface (API) that can be used to access external data provided by online service providers. Using mashups, key users and partners can integrate data provided by Web services with internal data from the SAP Business ByDesign system.

Web Service Authoring

A work center view in which key users can create and edit Web services.

Web service authorization

The process of authorizing users to access a business object by using a service operation.

Web Service Provisioning wizard

A guided activity for creating a Web service.

Web Services Inspection Language

A specification that supports you in searching for Web services that are available on a specific Web site. As a result the Web services and their endpoints are listed in a Web Service Inspection document.

window

The element of the graphical user interface used for displaying data and applications.

work center

An area where a user can perform tasks or functions that they are responsible for within a company. Examples of work centers are Purchase Requests and Orders and Personnel Administration.

work center proposals

A list of work centers proposed by the system based on the organizational assignment of the user.
work center view
A subdivision of a work center.

worklist
A list of items which may require user action.

XML Localization Interchange File Format
A standard XML-based file format used for exchanging bilingual content between systems during translation.

XML Schema Definition Language
An XML-based language that offers facilities for describing the structure and for constraining the contents of XML documents.
10 SAP - Copyrights and Trademarks

© 2012 SAP AG. All rights reserved.
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.
Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.
Microsoft, Windows, Excel, Outlook, PowerPoint, Silverlight, and Visual Studio are registered trademarks of Microsoft Corporation.
IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, z10, z/VM, z/OS, OS/390, zEnterprise, PowerVM, Power Architecture, Power Systems, POWER7, POWER6+, POWER6, POWER, PowerHA, pureScale, PowerPC, BladeCenter, System Storage, Storwize, XIV, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, AIX, Intelligent Miner, WebSphere, Tivoli, Informix, and Smarter Planet are trademarks or registered trademarks of IBM Corporation.
Linux is the registered trademark of Linus Torvalds in the United States and other countries.
Adobe, the Adobe logo, Acrobat, PostScript, and Reader are trademarks or registered trademarks of Adobe Systems Incorporated in the United States and other countries.
Oracle and Java are registered trademarks of Oracle and its affiliates.
UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.
Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems Inc.
HTML, XML, XHTML, and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.
Apple, App Store, iBooks, iPad, iPhone, iPhoto, iPod, iTunes, Multi-Touch, Objective-C, Retina, Safari, Siri, and Xcode are trademarks or registered trademarks of Apple Inc.
IOS is a registered trademark of Cisco Systems Inc.
RIM, BlackBerry, BBM, BlackBerry Curve, BlackBerry Bold, BlackBerry Pearl, BlackBerry Torch, BlackBerry Storm, BlackBerry Storm2, BlackBerry PlayBook, and BlackBerry App World are trademarks or registered trademarks of Research in Motion Limited.
Google App Engine, Google Apps, Google Checkout, Google Data API, Google Maps, Google Mobile Ads, Google Mobile Updater, Google Mobile, Google Store, Google Sync, Google Updater, Google Voice, Google Mail, Gmail, YouTube, Dalvik and Android are trademarks or registered trademarks of Google Inc.
INTERMEC is a registered trademark of Intermec Technologies Corporation.
Wi-Fi is a registered trademark of Wi-Fi Alliance.
Bluetooth is a registered trademark of Bluetooth SIG Inc.
Motorola is a registered trademark of Motorola Trademark Holdings LLC.
CompuTop is a registered trademark of CompuTop Wirtschaftsinformatik GmbH.
SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, SAP HANA, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.
Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective
logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company.

Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase Inc. Sybase is an SAP company.

Crossgate, m@gic EDDY, B2B 360°, and B2B 360° Services are registered trademarks of Crossgate AG in Germany and other countries. Crossgate is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.
11 Important Disclaimers on Legal Aspects

This document is for informational purposes only. Its content is subject to change without notice, and SAP does not warrant that it is error-free. SAP MAKES NO WARRANTIES, EXPRESS OR IMPLIED, OR OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Coding Samples

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages cause by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint where to find supplementary documentation. SAP does not warrant the availability and correctness of such supplementary documentation or the ability to serve for a particular purpose. SAP shall not be liable for any damages caused by the use of such documentation unless such damages have been caused by SAP’s gross negligence or willful misconduct.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with “you”, or a gender-neutral noun (such as “sales person” or “working days”) is used. If when referring to members of both sexes, however, the third person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.