Working with App Framework for SAP Business One, version for SAP HANA
# Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Textual cross-references to other documents.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Emphasized words or expressions.</td>
</tr>
<tr>
<td><strong>EXAMPLE</strong></td>
<td>Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><strong>&lt;Example&gt;</strong></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE</strong></td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>
# Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2013-12-9</td>
<td>First version</td>
</tr>
<tr>
<td>1.1</td>
<td>2014-9-15</td>
<td>Naming change of extreme app and extreme app framework</td>
</tr>
</tbody>
</table>
# Table of Contents

1. Introduction .............................................................................................................................................. 6

2. Getting Started ........................................................................................................................................ 8
   2.1 Prerequisites ......................................................................................................................................... 8
   2.2 Examples ........................................................................................................................................... 9
      2.2.1 Logging On to SAP Business One .......................................................................................... 9
      2.2.2 Hello World in HTML ........................................................................................................... 11
   2.3 Out-of-the-Box SAP Business One Apps ......................................................................................... 13

3. Developing the Apps ................................................................................................................................. 14
   3.1 App APIs ........................................................................................................................................... 14
      3.1.1 SAP Business One Web API for App ............................................................................... 14
      3.1.2 SAP Business One Mashup API for App .......................................................................... 21
   3.2 Creating User-Defined Stored Procedures ..................................................................................... 25

4. User Interface Guidelines ........................................................................................................................ 28

5. Implementing Lifecycle Management ..................................................................................................... 29
   5.1 Packaging the Apps ............................................................................................................................ 29
      5.1.1 Packaging App Artifacts in HANA ....................................................................................... 29
      5.1.2 Packaging Master Data of User-Defined Queries and Stored Procedures ........................ 29
      5.1.3 Creating Extension Registration Files ................................................................................... 30
   5.2 Deploying the Apps ............................................................................................................................. 31
      5.2.1 Deploying App Artifacts in SAP HANA ............................................................................... 31
      5.2.2 Registering with an Extension Registration File .................................................................... 31
      5.2.3 Deploying Apps to a Company ............................................................................................ 32
      5.2.4 Granting User Authorization ................................................................................................. 33
      5.2.5 Running the Apps ..................................................................................................................... 34

   6.1 Developing an App ATP ..................................................................................................................... 36
      6.1.1 Adding a System ....................................................................................................................... 36
      6.1.2 Creating Workspace ............................................................................................................... 36
      6.1.3 Creating an XS Project ........................................................................................................... 37
      6.1.4 Sharing a Project for SAP HANA XS .................................................................................... 38
      6.1.5 Creating an Application Descriptor File for Your Project .................................................. 39
      6.1.6 Creating an Application-Access File for Your Project ......................................................... 39
      6.1.7 Creating Source Files (HTML & CSS & JS) ......................................................................... 39
      6.1.8 Adding OData Service ............................................................................................................. 40
      6.1.9 Creating ATP Stored Procedures ........................................................................................... 40
      6.1.10 Creating User-Defined Stored Procedures .......................................................................... 40
      6.1.11 Testing the App ....................................................................................................................... 41
   6.2 Packaging App ATP ............................................................................................................................. 42
      6.2.1 Exporting Delivery Unit ........................................................................................................... 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.2</td>
<td>Using SAP Business One DI API to Package Master Data of UDQ and UDSP</td>
<td>44</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Creating Extension Registration Files</td>
<td>45</td>
</tr>
<tr>
<td>6.3</td>
<td>Deploying App ATPs</td>
<td>45</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Importing Delivery Units</td>
<td>45</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Registering Apps in SAP Business One</td>
<td>45</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Assigning Apps to Your Company</td>
<td>46</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Assigning User Permissions</td>
<td>46</td>
</tr>
<tr>
<td>6.3.5</td>
<td>Activating OData Service for the Current Company</td>
<td>47</td>
</tr>
<tr>
<td>6.3.6</td>
<td>Running Your DI API Add-on</td>
<td>47</td>
</tr>
<tr>
<td>6.3.7</td>
<td>Running Your Apps</td>
<td>47</td>
</tr>
</tbody>
</table>
1 Introduction

This document describes how to work with the App Framework for SAP Business One, version for SAP HANA. To do so, you should have a basic familiarity with software development, web services, and SAP HANA.

The App Framework for SAP Business One, version for SAP HANA is powered by SAP HANA technology and SAP HANA extended application services (SAP HANA XS). It enables SAP partners to build analytics-based lightweight apps on the SAP HANA XS engine. SAP HANA XS is a lightweight application server embedded directly in the SAP HANA database system. Using the features provided by SAP HANA XS, you can build server applications that run on SAP HANA without the need for an additional application server.

A typical high-level flow of procedures for working with the App Framework for SAP Business One, version for SAP HANA is as follows:

1. Develop your app for the version for SAP HANA.
2. Commit and activate your project in SAP HANA’s XS engine, and view the results in a web browser.
3. Package and deploy your app.

The following figure shows the architecture of the app framework for SAP Business One, version for SAP HANA.

In this version, the app framework for SAP Business One, version for SAP HANA provides the following features:

- **Web API**
  - **Login Service**: RESTful web service that provides authentication verification. It logs in the business user.
  - **Query Service**: RESTful web service that executes predefined user-defined queries (UDQ) and user-defined stored procedures (UDSP).
Environment Service:

- **Environment Service**: RESTful web service that returns the SAP Business One related environment variables for the current user.

**Lifecycle Management**

- You can package your apps using the Extension Registration Data Generator tool.
- You can manage the stored procedure master data using DI API.
- You can work with the App Center for SAP Business One, a widget that displays all registered apps. The widget lets you register the apps, assign/unassign companies, and manage user authentication.

**Single Sign On (SSO) in SAP Business One**

You can use the apps in the SAP Business One client without logging in to the apps explicitly.

### Terms and Definitions

The following terms are used in this document.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP HANA</td>
<td>SAP High-Performance Analytical Appliance</td>
</tr>
<tr>
<td>SAP HANA XS</td>
<td>A lightweight application server embedded directly in the SAP HANA Database system that provides access to the SAP HANA database using a consumption model exposed via HTTP.</td>
</tr>
<tr>
<td>Apps Available with SAP Business One, version for SAP HANA</td>
<td>A lightweight application with real time analytics capability.</td>
</tr>
<tr>
<td>DI API</td>
<td>SAP Business One SDK Data Interface Application Programming Interface</td>
</tr>
</tbody>
</table>

### Related Documentation

The documents listed in the table are referred to in this document.

<table>
<thead>
<tr>
<th>Document</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>The document includes the installation information of the App Framework for SAP Business One, version for SAP HANA.</td>
<td></td>
</tr>
<tr>
<td>SDK online help file SDK_EN.chm</td>
<td><a href="http://service.sap.com/smb/sbocustomer/documentation">http://service.sap.com/smb/sbocustomer/documentation</a> and choose Release Family 9.0 → SDK and Custom Development Available also on the SAP Business One product DVD and in the download package from SAP Service Marketplace</td>
</tr>
</tbody>
</table>
2 Getting Started

The information in this section explains what you need to do to set up your SAP Business One app development environment and, with the help of simple examples, takes you through the basic scenarios you will encounter when developing applications for SAP Business One.

2.1 Prerequisites

- You have installed SAP Business One Server Tools.
  SAP Business One App Framework is one of the components of SAP Business One Server Tools. After you install SAP Business One Server Tools, the application deploys all App Framework artifacts into SAP HANA and configures SAP HANA XS with embedded mode and SSL mode.
  The port number is configured to 43xx, where xx represents the SAP HANA instance number.
- You have installed the SAP HANA server.
- You have installed the 32-bit version of the SAP HANA client for Linux. The installation path must be the default path `.../usr/sap`.
  
  **Note**
  To use the apps, you must also install the 64-bit version of the SAP HANA database client.
- You have installed SAP HANA studio.
- You have created an SAP HANA development user.
  Every developer needs to have a database user to be able to update and retrieve content from the database. Perform the following steps on the HANA system as a user with system privileges (for example: SYSTEM user):
  1. Open HANA Studio Navigator view.
  2. Choose **SYSTEM → Catalog → Authorization → Users → [context menu] → New User**.
  3. Enter the user’s name and an initial password.
  4. Grant the following roles:
     - CONTENT_ADMIN
     - MODELING
  5. Choose deploy (F8).
- You have installed SAP Business One SDK and SDK Tools.
  SAP Business One SDK is used for the packaging and deployment of some apps artifacts, so you need to install it before implementing the lifecycle management of the app. The `ExtensionRegDataGen.exe` inside SDK Tools is used for app registration file creation.
2.2 Examples

To help you get started with app development in the App Framework for SAP Business One, version for SAP HANA, we provide the following **Hello World** examples.

### 2.2.1 Logging On to SAP Business One

Before calling any of the SAP Business One App services, you must log on to SAP Business One. When an app is running inside the SAP Business One browser widget, the app itself does not need to log on explicitly, because SAP Business One performs single sign-on for all apps embedded inside the browser widget. However, when an app is under development, a standalone browser is more convenient for debugging, and so SAP Business One provides the login service to enable app logons to the SAP Business One client from outside of SAP Business One.

The following example demonstrates how an app logs on to SAP Business One through the login service.

#### Procedure

1. **In your workspace, create package** `sap.test.helloworld`.
2. **Under** `/sap/test/helloworld`, **create** `.xsapp` and `.xsaccess` files.
3. **Create** `login.html`, **open it**, and enter the following code:

```html
<!DOCTYPE html>
<html>
<head>
<script src="jquery.js"></script>
<script>
$(function () {
    /*Login service sample code*/
    $('#login').click(function () {
        var button = $(this);
        $.ajax{
            type: "POST",
            url: "../../../platform/login",
            data: {
                "company": $('#company').val(),
                "username": $('#b1user').val(),
                "password": $('#b1pwd').val(),
                "language": $('#b1language').val()
            },
            error: function (xhr, status, error) {
                window.alert("login failed: " + xhr.responseText);
            },
        };
    };
})(jQuery);
</script>
</head>
<body>
</body>
</html>
```

This example demonstrates the login service by using jQuery to define a click event for a login button. When the button is clicked, an AJAX request is sent to the login service with the company, username, password, and language details. If the login fails, a warning message is displayed to the user.
success: function () {
    window.alert("login successfully.");
}];
});
});
});
</script>
</head>
<body>
<div id="container">

<!--Login Service-->
<h1>Login Service</h1>
<p>
<label for="company" style="display: block">B1 Company: </label>
<input type="text" id="company" value="SBODEMOUS" />
<br />
<label for="b1user" style="display: block">B1 User: </label>
<input type="text" id="b1user" value="manager" />
<br />
<label for="b1pwd" style="display: block">B1 Password: </label>
<input type="password" id="b1pwd" value="manager" />
<br />
<label for="b1language" style="display: block">B1 Language: </label>
<input type="text" id="b1language" value="en-US" />
<br />
<input type="button" value="Login" id="login" />
</p>
</div>
</body>
</html>

4. Save, commit and activate your project.
   To verify the result, open your web browser and in the address bar, enter the URL https://<xs_host:port>/sap/test/helloworld/login.html. The successful result appears.
2.2.2 Hello World in HTML

The following example demonstrates how an app gets the environment settings (such as `TimeTemplate`, `SystemCurrency`, `Country`, and so on) of the currently logged-on companies.

Procedure

1. In your workspace, create package `sap.test.helloworld`.
2. Under `/sap/test/helloworld`, create `.xsapp` and `.xsaccess` files.
3. Create `env.html`, open it, and enter the following code:

   ```html
   <!DOCTYPE html>
   <html>
   <head>
   <script src="jquery.js"></script>
   <script>
   $(function () {
   /*Environment service sample code*/
   ```
$("#env").click(function () {
    var button = $(this);
    button.attr("disabled", "disabled");
    $.ajax({
        type: "GET",
        url: ".\..\..\..\platform\env",
        dataType: "json",
        error: function (xhr, status, error) {
            window.alert("env failed: " + xhr.responseText);
        },
        success: function (data) {
            window.alert("env successfully.");
            var table = $("#env-result");
            table.html(""");
            for (k in data) {
                if (Object.prototype.hasOwnProperty.call(data, k)) {
                    table.append("<tr><td>" + k + "</td><td>" + data[k] + "</td></tr>");
                }
            }
        }
    });
});
</script>
</head>
<body>
<div id="container">
    <!--Environment Service-->
    <h1>Environment Service</h1>
    <div>
        <table id="env-result">
        </table>
        <input type="button" value="Get Environment" id="env" />
        <br />
        <label for="language" style="display: block">Language: </label>
        <input type="text" id="language" value="en-GB" />
        <br />
        <input type="button" value="Set Language" id="btnlanguage" />
    </div>
</div>
4. Save, commit, and activate your project.

**Note**
Before running this sample, make sure your browser or the SAP Business One client has been authenticated by the login service.

To verify the result, open your web browser and in the address bar, enter the URL https://<xs_host:port>/sap/test/helloworld/env.html. The successful result appears.

## Environment Service

<table>
<thead>
<tr>
<th>Environment Service</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxCharOfMonth</td>
<td>null</td>
</tr>
<tr>
<td>TimeTemplate</td>
<td>0</td>
</tr>
<tr>
<td>DateTemplate</td>
<td>3</td>
</tr>
<tr>
<td>DateSeparator</td>
<td>/</td>
</tr>
<tr>
<td>DecimalSeparator</td>
<td>.</td>
</tr>
<tr>
<td>ThousandsSeparator</td>
<td>.</td>
</tr>
<tr>
<td>AccuracyOfQuantities</td>
<td>3</td>
</tr>
<tr>
<td>LocalCurrency</td>
<td>$</td>
</tr>
<tr>
<td>SystemCurrency</td>
<td>$</td>
</tr>
<tr>
<td>DisplayCurrencyOnTheRight</td>
<td>Y</td>
</tr>
</tbody>
</table>

### 2.3 Out-of-the-Box SAP Business One Apps

SAP provided the following out-of-the-box Apps for you:

  After you fill in all the fields and choose **Login**, the application logs in both HANA XS and SAP Business One, and redirect to the site you desire (in this case, is [https://10.58.1.134:4300/your/company/myapp/](https://10.58.1.134:4300/your/company/myapp/)).

  This demo is designed for your reference of coding your own App. The UI API sample is located in the SDK sample folder.
3 Developing the Apps

3.1 App APIs

App Framework for SAP Business One, version for SAP HANA provides a number of resources for developers, third parties, and app enthusiasts. Most of the SAP Business One business data, including the semantic layer, can be read and written by the APIs.

3.1.1 SAP Business One Web API for App

App web API is a set of RESTful APIs that exposes the SAP Business One data. The target consumers should be browsers, mobile apps, and all REST-compatible clients.

App web API requires authentication before any further calls. For information on how to log in the App Framework, see the Login Service section.

3.1.1.1 Login Service

Login service provides authentication verification. It logs in the business user, and creates a session for the business user.

REQUEST

POST /sap/sbo/platform/login
Host: [XS Host]
Content-Type: application/x-www-form-urlencoded
Authorization: Basic {Base64 encoded HDB username and password}

"company": "company name",
"username": "b1 user name",
"password": "b1 user password"
"language": "preferred language"

SUCCESSFUL RESPONSE

HTTP/1.1 200 OK
Cookie: xsSessionId=GUID

UNSUCCESSFUL RESPONSE

HTTP/1.1 {various status code}
Content-Type: application/json

{
   "error": "reason of failure"
}

Note
Currently SAP HANA XS does not support integrated authentication. Hence, SAP HANA database credentials, as well as SAP Business One credentials, must be provided at the same time to pass the login procedure.

3.1.1.2 Query Service

The query service allows you to execute specific user-defined queries (UDQ) or user-defined stored procedures (UDSP).

Note
Only authorized users can consume this service. The execution of UDQs and UDSPs is under the control of user authorization for the corresponding UDQ/UDSP master data. For more information, see the Query Manager section in SAP Business One online help.

REQUEST

POST /sap/sbo/platform/query
Host: [XS Host]
Cookie: xsSessionId=GUID

{
   "type": "sql or sp",
   "category": "category name",
   "name": "UDQ or UDSP name",
   "param": ["Hello World", (string type param 1)
   2276, (integer type param 2)
   "U11TVEVNOmlhbFn2Xl=", (blob type param 3, base64 encoded)
   null, (null for param 4)
   "2013-07-19 23:15:03.045" (timestamp type param 5) ...],
   "format": "CondenseJSON or JSON" (optional)
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Indicates whether the query service executes a UDQ(sql) or a UDSP(sp).</td>
</tr>
<tr>
<td>Category</td>
<td>The query service finds UDQ/UDSP according to the given category name and query name. This parameter is mapped to the database field OQCN.CatName.</td>
</tr>
<tr>
<td>Name</td>
<td>The query service finds UDQ/UDSP according to the given category name and query name. This parameter is mapped to the database field OUQR.QName.</td>
</tr>
<tr>
<td>Param</td>
<td>A JavaScript array of input, output, and inout parameters for executing the specific UDQ/UDSP.</td>
</tr>
<tr>
<td>Format</td>
<td>Indicates the output data format that the query service is going to apply. This parameter is optional. The default value is CondenseJSON. This format generates JSON data as concisely as possible. The following code shows the difference between CondenseJSON and JSON formats:</td>
</tr>
</tbody>
</table>

```
{  
  "meta": [{  
    "index": 0,  
    "name": "WAREHOUSE",  
    "type": "NVARCHAR "  
  },  
  {  
    "index": 1,  
    "name": "QTY1",  
    "type": "INTEGER"  
  },  
  {  
    "index": 2,  
    "name": "QTY2",  
    "type": "INTEGER"  
  },  
  {  
    "index": 3,  
    "name": "QTY3",  
    "type": "INTEGER"  
  }  
}
```
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
|          | }},
|          | "data": [["General Warehouse",
|          | 980,
|          | 980,
|          | 980],
|          | 
|          | 
|          | ["West Cost Warehouse",
|          | 0,
|          | 0,
|          | 0],
|          | ["Dropship Warehouse",
|          | 0,
|          | 0,
|          | 0],
|          | ["Consignment1 Warehouse",
|          | 0,
|          | 0,
|          | 0]],
|          | "param": []
|          | }
|          | • **JSON**
|          | {
|          | "meta": [[{
|          | "index": 0,
|          | "name": "WAREHOUSE",
|          | "type": "NVARCHAR"
|          | }
|          | ,
|          | {
|          | "index": 1,
|          | "name": "QTY1",
|          | "type": "INTEGER"
|          | }
|          | ,
|          | {
|          | "index": 2,
|          | "name": "QTY2",
|          | "type": "INTEGER"
|          | }
|          | ,
|          | {
|          | "index": 3,
|          | "name": "QTY3",
|          | ]
|          | ]}
JSON data is compatible with SAP UI5. For more information on how SAP UI5 consumes the query service, see QueryServiceForUI5Sample.html from the zipped package.

SUCCESSFUL RESPONSE

HTTP/1.1 200 OK
Content-Type: application/json

{
    "param": [output params in array],
    "meta": [metadata of the response result set in array],
}
UNSUCCESSFUL RESPONSE

HTTP/1.1 {various status code}
Content-Type: application/json

{
   "error": "reason of failure"
}

Query Service Limitations

- The query service cannot handle the nCLOB data type if the data size is larger than 500 bytes. This is due to a limitation of SAP HANA Platform Edition 1.0 SPS 05 Rev52.
- The query service input payload has a size limitation of 20M bytes. If the payload exceeds the limitation, you get a 400 exception.
- The query service output data set has a maximum of 20 results. The excess results are truncated without error or warning message.
- The query service output data set has a maximum of 1000 lines for each result. The excess lines are truncated without error or warning message. We recommend that you work with paging in queries.
- The query service output data set has a maximum data size of 20M bytes for each BLOB data type. The excess is replaced with an error message.
- The query service returns error code 500 with a non-meaningful HTML response error page if it encounters an internal out-of-memory exception. This is due to a limitation of SAP HANA Platform Edition 1.0 SPS 05 Rev52.

3.1.1.3 Environment Service

The environment service gets the SAP Business One related environment variables for the current user. The source table in SAP Business One is OADM.

1. Note

   Only the logged-on user can consume this service.

The following table lists the DB fields and the response fields that appear in the environment service result:

<table>
<thead>
<tr>
<th>Response Fields</th>
<th>DB Fields</th>
<th>DB Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxCharOfMonth</td>
<td>CharMonth</td>
<td>OADM</td>
</tr>
<tr>
<td>TimeTemplate</td>
<td>TimeFormat</td>
<td>OADM</td>
</tr>
<tr>
<td>Response Fields</td>
<td>DB Fields</td>
<td>DB Table</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>DateTemplate</td>
<td>DateFormat</td>
<td>OADM</td>
</tr>
<tr>
<td>DateSeparator</td>
<td>DateSep</td>
<td>OADM</td>
</tr>
<tr>
<td>DecimalSeparator</td>
<td>DecSep</td>
<td>OADM</td>
</tr>
<tr>
<td>ThousandsSeparator</td>
<td>ThousSep</td>
<td>OADM</td>
</tr>
<tr>
<td>AccuracyofQuantities</td>
<td>QtyDec</td>
<td>OADM</td>
</tr>
<tr>
<td>LocalCurrency</td>
<td>MainCurncy</td>
<td>OADM</td>
</tr>
<tr>
<td>SystemCurrency</td>
<td>SysCurreny</td>
<td>OADM</td>
</tr>
<tr>
<td>DisplayCurrencyontheRight</td>
<td>CurOnRight</td>
<td>OADM</td>
</tr>
<tr>
<td>PriceAccuracy</td>
<td>PriceDec</td>
<td>OADM</td>
</tr>
<tr>
<td>QueryAccuracy</td>
<td>QueryDec</td>
<td>OADM</td>
</tr>
<tr>
<td>PercentageAccuracy</td>
<td>PercentDec</td>
<td>OADM</td>
</tr>
<tr>
<td>TotalsAccuracy</td>
<td>SumDec</td>
<td>OADM</td>
</tr>
<tr>
<td>RateAccuracy</td>
<td>RateDec</td>
<td>OADM</td>
</tr>
<tr>
<td>MeasuringAccuracy</td>
<td>MeasureDec</td>
<td>OADM</td>
</tr>
<tr>
<td>CompanyName</td>
<td>CompnyName</td>
<td>OADM</td>
</tr>
<tr>
<td>Country</td>
<td>Country</td>
<td>OADM</td>
</tr>
<tr>
<td>State</td>
<td>State</td>
<td>OADM</td>
</tr>
<tr>
<td>Language</td>
<td>UserPrefs</td>
<td>OUSR</td>
</tr>
</tbody>
</table>

**REQUEST**

GET /sap/sbo/platform/env
Host: [XS Host]
Cookie: xsSessionId=GUID

**SUCCESSFUL RESPONSE**

HTTP/1.1 200 OK
Content-Type: application/json

```json
{
  "MaxCharOfMonth": "...",
  "TimeTemplate": "...",
  "DateTemplate": "...",
```
3.1.2 SAP Business One Mashup API for App

SAP Business One Mashup API for App is a front-end JavaScript library that enables you to interact with SAP Business One client from HTML based App.

1 Note

Before using the Mashup APIs, you must reference `webbridge.js` into the web pages.

```html
<script src="/sap/sbo/platform/js/webbridge.js"></script>
```
3.1.2.1 Opening Messages in SAP Business One

Namespace: sap.sbo.webbridge

You can open message windows in SAP Business One client by using the following functions:

• showError(message: string, ...objs: any[]): void
  This function displays an error level message in SAP Business One client.

• showNote(message: string, ...objs: any[]): void
  This function displays a note level message in SAP Business One client.

• showWarning(message: string, ...objs: any[]): void
  This function displays a warning level message in SAP Business One client.

• showSuccess(message: string, ...objs: any[]): void
  This function displays a success level message in SAP Business One client.

The functions are designed in the same pattern, which accepts one message with format and variadic objects to apply the format. The functions have no return value and will not throw any exception.

Sample Code

```javascript
sap.sbo.webbridge.showError("Hello {0}, this is business one, today is {1}.", "your name", new Date());
sap.sbo.webbridge.showSuccess("This is a success message");
```

3.1.2.2 Opening Forms in SAP Business One

Namespace: sap.sbo.webbridge

You can use the function openForm(table: BoTable, key: string): void to open a specific SAP Business One form by the given table name and the primary key value.

This function accepts two parameters:

• **Table** is of string type. You can pass the predefined values in the enumeration object BoTable as well as any string value that represents the table name within the database.

• **Key** is also of string type, which is the value of the primary key in the table. When you specify a key that meets the corresponding record in SAP Business One, the application opens the window and navigates to the record.

The function has no return value and will not throw any exception.

Sample Code

```javascript
var BoTable = sap.sbo.webbridge.BoTable;
sap.sbo.webbridge.openForm(BoTable.Invoice, "1");
sap.sbo.webbridge.openForm("UDT1", "My PK value");
```

Refer to the object enum BoTable for the table name:

```javascript
export enum BoTable {
    AgentPerson = <any>'OAGP',
    BillOfExchange = <any>'OBOE',
}```
BillOfExchangeTransaction = <any>'OBOT',
BPBankAccount = <any>'OCRB',
BudgetSystem = <any>'OBGD',
BusinessPartner = <any>'OCRD',
CashDiscount = <any>'OCDC',
CentralBankIndicator = <any>'OCBI',
CheckForPayment = <any>'OCHO',
ConfirmationDocument = <any>'OWDD',
ConfirmationLevel = <any>'OWST',
ConfirmationTemplates = <any>'OWTM',
ContactWithCustAndVend = <any>'OCLG',
ContractTemplate = <any>'OCTT',
CreditCards = <any>'OCRC',
DeliveryNotes = <any>'ODLN',
DeliveryNotesReturns = <any>'ORDN',
DeliveryTypes = <any>'OSHP',
Deposit = <any>'ODPS',
DiscountCodes = <any>'ODSC',
DunningTerms = <any>'ODUT',
Employee = <any>'OHEM',
ExpensesDefinition = <any>'OEXD',
FileFormat = <any>'OFRM',
FinancePeriod = <any>'OFPR',
GLAccounts = <any>'OACT',
GoodsIssue = <any>'OIGE',
GoodsReceipt = <any>'OIGN',
GoodsReceiptPO = <any>'OPDN',
GoodsReturns = <any>'ORPD',
GoodsShipment = <any>'OGSP',
HolidaysTable = <any>'OHLD',
ImportFile = <any>'OIPF',
Indicator = <any>'OIDC',
InstallBase = <any>'OINS',
Invoice = <any>'OINV',
InvoiceCreditMemo = <any>'ORIN',
ItemBatchNumbers = <any>'OIBT',
ItemGroups = <any>'OITB',
Items = <any>'OITM',
JournalPosting = <any>'OJDT',
JournalVoucher = <any>'OBTF',
LoadingFactors = <any>'OOCR',
Order = <any>'ORDR',
PaymentBlock = <any>'OPYB',
PaymentMethod = <any>'OPYM',
PaymentTermsTypes = <any>'OCTG',
PeriodIndicator = <any>'OPID',
PickList = <any>'OPKL',
PredatedDeposit = <any>'ODPT',
PredefinedText = <any>'OPDT',
ProductionOrder = <any>'OWOR',
ProductTree = <any>'OITT',
ProjectCodes = <any>'OPRJ',
PurchaseInvoice = <any>'OPCH',
PurchaseInvoiceCreditMemo = <any>'ORPC',
PurchaseOrder = <any>'OPOR',
Quotation = <any>'OQUT',
Receipt = <any>'ORCT',
SalesForecast = <any>'OFCT',
SalesOpportunity = <any>'OOPR',
SalesTaxCodes = <any>'OSTC',
SerialNumbersForItems = <any>'OSRI',
ServiceCall = <any>'OSCL',
ServiceCallSolution = <any>'OSLT',
ServiceContract = <any>'OCTR',
SpecialPrices = <any>'OSP',
StockRevaluation = <any>'OMRV',
StockTransfers = <any>'OWTR',
StockTransfersRequest = <any>'OWTQ',
Territory = <any>'OTER',
TransactionTemplates = <any>'OTRT',
User = <any>'OUSR',
UserDefaults = <any>'OUDG',
VatIndicator = <any>'OIND',
VendorPayment = <any>'OVPM',
Warehouses = <any>'OWHS',
WithHoldingTax = <any>'OWHT',
WorkInstructions = <any>'OWKO',
}
3.2 Creating User-Defined Stored Procedures

You can define and deploy your own stored procedures and maintain the master data of the procedures. After you create your own stored procedures in the SAP HANA database, you should create an entry in the Query Manager window of SAP Business One.

1. From the Tools menu of SAP Business One, choose Queries → Query Manager. The Query Manager window appears.
2. Choose the New Stored Procedure button. The User Defined Stored Procedure window appears.
3. Specify an alias for the stored procedure.

![User Defined Stored Procedure window](image)

**Note**

Naming conventions for UDQ and UDSP:
- For the category name of UDQ/UDSP, use the partner namespace as a prefix, followed by an underscore (_) and the category sub name. For example, SAP_MyCategory1 (The namespace is specified in the extension registration tool).
- For the UDSP alias, use the partner namespace as a prefix, followed by an underscore (_) and the UDSP sub name. For example, SAP_MyStoredProcedure1.
- The length of a UDQ name and a UDSP alias is restricted to 100 characters.
- The length of a stored procedure name is restricted to 256 characters.
4. Choose the Save button. The Save Query window appears.
5. Save the query in the *General* category and choose **OK**.

---

**Note**

To manage the authorization groups for the categories, choose the **Manage Categories** button.

For more information, see the *Create/Edit Categories Window* section in SAP Business One online help.
To manage the user authorization, from the SAP Business One Main Menu, choose Administration → System Initialization → Authorizations.

For more information, see the Authorizations section in SAP Business One online help.
4 User Interface Guidelines

The user interface guidelines provide best practices for designing the user interface of your apps for the version for SAP HANA. For more information, see http://help.sap.com/download/multimedia/sapb1_xapp/uiguideline/sap/sbo/guideline/.

By following these guidelines, developers, partners, and product managers can apply the most updated GUI design rules and recommendations for your application.
5 Implementing Lifecycle Management

As an app developer, you check out design-time content from the SAP HANA repository, edit a copy of the checked-out artifact in the local file system on your personal computer (PC), deploy it into a productive/development system, and run it in a SAP Business One client.

The following steps are a brief, high-level overview of the development lifecycle for design-time content:

1. Packaging an app.
   Export the package, containing the design-time artifacts and the master data definition, from a development system.
2. Deploying an app.
   Import the package into a productive/development system, and run it in SAP Business One.

5.1 Packaging the Apps

5.1.1 Packaging App Artifacts in HANA

Apps development on SAP HANA requires a server-centric lifecycle for design-time objects, which are the development artifacts that you store in the SAP HANA repository. In SAP HANA, the delivery unit (DU) is a collection of packages that are to be transported together. It is the vehicle that lifecycle management (LCM) uses to ship one or more software components from SAP (or an SAP partner) to our customer.

For more information, see the following:
- Section 5.2 of SAP HANA Developer Guide to set up a delivery unit.
- Section 14.3 of SAP HANA Developer Guide to export your app artifacts into a delivery unit.

The App artifacts include your application package inside SAP HANA XS Engine (for example, HTML, JavaScript, CSS, OData definition files and so on.) and any other implementation artifacts for your app content inside SAP HANA.

5.1.2 Packaging Master Data of User-Defined Queries and Stored Procedures

The master data of user-defined queries and stored procedures used in an app, are important for the query/stored procedure access from the App Framework, SAP Business One user authorization, and extension lifecycle management. You must develop an add-on through SAP Business One DI API to package the master data.

Refer to the following example to manage a user-defined stored procedure using the UserQueries object.
SAPBobsCOM.UserQueries userQ =
(SAPBobsCOM.UserQueries) company.GetBusinessObject(SAPBobsCOM.BoObjectTypes.oUserQueries);
userQ.QueryCategory = -1;
userQ.QueryType = SAPBobsCOM.UserQueryTypeEnum.uqtStoredProcedure;
userQ.ProcedureAlias = "TestSP";
userQ.ProcedureName = "\"MySPName\"";
int iRet = userQ.Add();

For more information about the UserQueries object, refer to the SAP Business One SDK Help Center.
You can also use the add-on to package stored procedures to be created in targeting company databases. For example, if you have a stored procedure named MySP, you can package its definition as a script file, read it in the add-on, and create the stored procedure in the targeting company via the DI API Recordset object.

5.1.3 Creating Extension Registration Files

To create a registration file for the apps, run ExtensionRegDataGen.exe from SAP Business One SDK tools.
The registration data file (.ard extension) enables you to register your app with the SAP Business One application.
To create an extension registration file, perform the following steps:
1. From the SAP Business One SDK Tools folder, run ExtensionRegDataGen.exe.
   The ExtensionRegDataGen.exe file is typically located at: ..\Program Files (x86)\sap\SAP Business One SDK\Tools\ExtensionRegDataGen.
2. Specify the partner information: Name, Namespace, and Contact Info.
3. Specify the extension information: Extension Type, Name, Package, and Version.

   Note
   The naming convention for your package is *.*. If the package hierarchy is sap.sbo.atp in SAP HANA, the package name is this, and the corresponding URL will be https://host:port/sap/sbo/atp.
4. Choose Export to generate the .ard file.
5. If you want to edit an .ard file, choose Import.
   The information in your .ard file is loaded in the fields. You can edit the fields and choose Export to generate
   the updated .ard file.

5.2 Deploying the Apps

To deploy an app, perform the following steps:
1. Deploy the app artifacts into SAP HANA.
2. Register the app to SAP Business One.
3. Deploy the app to companies.
4. Authorize users in the companies for the app.
5. Run the app in SAP Business One.

5.2.1 Deploying App Artifacts in SAP HANA

Importing a delivery-unit archive is the mechanism SAP HANA uses to enable you to transport delivery units
between systems. To import the delivery unit created in section 5.1.1 into your target system, see Section 14.4 of
SAP HANA Developer Guide.

Note
It is not recommended that you change the packages sap.sbo.platform and sap.sbo.lcm.

5.2.2 Registering with an Extension Registration File

To register an app to SAP Business One, perform the following steps:
1. In the SAP Business One client, from the menu bar, choose Tools → App Center.
   The App Center window appears.
2. Choose the Registration button.
   The Register App window appears.
3. Choose the registration file of your app (created in step 5.1.3) and choose OK.
   The app is registered, and appears on the left panel as Unassigned Apps.

Note
Access to App Center is under SAP Business One user authorization. Only users authorized to access the
window can open it. You can grant authorizations in the Authorizations window (from the SAP Business
One Main Menu, choose Administration → System Initialization → Authorizations → General
Authorizations). For more information about user authorization, see SAP Business One online help.
5.2.3 Deploying Apps to a Company

The deployment of an app to a company includes the following steps:

1. Deploy the master data of the user-defined query/stored procedure.
   To deploy user-defined query/stored procedure master data in this company, you must run the add-on created in section 5.1.2.

2. Assign the app to a company.
   To assign an app to a company, drag the app from the Unassigned App list to the Active App area. This is a sign that the app is accessible in this company.

   **Note**

   If an app is not assigned to the company, you may still access the app from this company. However, the correctness of running the unassigned apps is not ensured under the app lifecycle management.
5.2.4 Granting User Authorization

Choose the User Authorization button to authorize users to access the apps assigned to the company.

- Super users by default have authorization to apps. You cannot remove their authorization.
- Non-super users can be configured with specific authorization.

If a user is selected, the user can access the apps in the SAP Business One client; otherwise, the user is blocked from accessing them.

\[\text{Note}\]

If a user authorization is changed, the updated status takes effect at the next logon of the corresponding user.

\[\text{Note}\]

App Center is an app repository implemented by SAP Business One. To obtain permission to access the center, you must have the authorization for the window and the apps.
5.2.5 Running the Apps

You can run your apps from your web browser, SAP Business One Browser Widget, or your own add-on.

5.2.5.1 Running Apps in Web Browser

To run an app in the web browser, perform the following steps:

2. Fill in all the fields and choose Login.
   The application logs in both HANA XS and SAP Business One, and redirect to the site you desire (in this case, is https://10.58.1.134:4300/your/company/myapp/)

5.2.5.2 Running Apps in SAP Business One Browser Widget

To run an app in SAP Business One, perform the following steps:

1. Switch to cockpit view and open a browser widget. For more information, see the Working with the Cockpit section in SAP Business One online help.
2. Choose the Settings button in the top-right of the screen.
   The Browser Widget - Setting window appears.
3. In the Type field, from the drop-down box, select App.
4. In the App field, select the app or enter the URL of the app.
   Note
   The URL of an app can be transformed from the package path of the app. For example, if an app package is sap.sbo.xapp1, its URL is https://hostname:4300/sap/sbo/xapp1.
5. Choose OK.
   The app is running in the browser widget.

5.2.5.3 Running Apps by Adding a WebBrowser Object into your Add-On via UI API

To run an app from your add-on, you need to use the newly added UI API object WebBrowser. The WebBrowser object enables you to place a web browser in your add-on form.
Sample Code

SAPbouiCOM.Item oItem = oFirstForm.Items.Add("WebBrowser", SAPbouiCOM.BoFormItemTypes.it_WEB_BROWSER);
oItem.Left = 20;
oItem.Top = 20;
oItem.Width = 200;
oItem.Height = 200;
SAPbouiCOM.WebBrowser oWebBrowser = (SAPbouiCOM.WebBrowser)oItem.Specific;
//Open a WebPage
oWebBrowser.Url = "https://hostname:4300/sap/sbo/xappl";
6 Tutorial: Step by Step Building the Apps: Building an App ATP

6.1 Developing an App ATP

This sample guilds you to build an app ATP step by step. The app ATP is using part of the ATP functionality in SAP Business One. For more information about the ATP functionality, see Advanced Available to Promise (ATP) of the SAP Business One online help.

You can get the sample source code from the zipped package atp.zip.

6.1.1 Adding a System

1. In the SAP HANA studio, open the SAP HANA Development perspective.
2. Choose the Navigator view, right-click anywhere in the view and select Add System.

3. In the pop-up window, enter the following fields for the SAP HANA system:
   o Server name
   o Instance number on that server
   o A display name for this system
4. Choose Next.
5. Enter a user name and password for the connection, and choose Finish.
   The newly added system appears in the Navigator view.

6.1.2 Creating Workspace

1. In the SAP HANA studio, open the SAP HANA Development perspective.
2. Choose the SAP HANA Repositories view.

3. From the top right-hand corner of the SAP HANA Repositories view, choose the New Repository Workspace button.

4. Specify the workspace details.

   In the Create Workspace window, enter the following information and choose Finish:
   
   o Specify the SAP HANA system for which you want to create a new workspace.
   
   o Enter a workspace name, which can be anything you like, for example, the name of the SAP HANA system where the repository is located.
   
   o Specify where the workspace root directory should be located on your local file system, for example, C:sers\username\workspaces.

   ![Create Workspace Window](image)

   The new workspace appears in the SAP HANA Repositories view.

6.1.3 Creating an XS Project

1. In the SAP HANA studio, open the SAP HANA Development perspective.

2. Choose the Project Explorer view.

3. Right-click the white space in the Project Explorer view and choose New → Project....

4. Under the SAP HANA Development perspective, select XS Project, and choose Next.

5. Enter the project name and location. For example, here we use "atp" as the project name.
6. To create the new project, choose Finish.
The new project is displayed in the Project Explorer view.

### 6.1.4 Sharing a Project for SAP HANA XS

1. In the SAP HANA studio, open the SAP HANA Development perspective.
2. Choose the Project Explorer view.
3. Right-click the project you want to share and, choose Team → Share Project…
4. In the Share project window, select the repository workspace where the project should be located and specify the package with which you want to associate the shared project.
   The Share project window displays the suggested location for the shared project in the New project location field. The default location is the name of the workspace with the name of the project you want to share. Choose Browse... to locate the package with which you want to associate the shared project. The selected package is displayed in the Repository package field.

![Share project window](image)

5. To complete the project-sharing procedure, choose Finish.
6.1.5 Creating an Application Descriptor File for Your Project

1. In the SAP HANA studio, open the SAP HANA Development perspective.
2. In the Project Explorer view, right-click your project and choose New → File to create a new .xsapp file.
3. Enter the name of the .xsapp file and choose Finish.
   The content of the file is empty.
4. Commit and activate the .xsapp file in the SAP HANA repository.
   To commit the .xsapp file, right-click it and choose Team → Commit.
   To activate the .xsapp file, right-click it and choose Team → Activate.

6.1.6 Creating an Application-Access File for Your Project

1. In the SAP HANA studio, open the SAP HANA Development perspective.
2. In the Project Explorer view, right-click your project and choose New → File to create a new .xsaccess file.
3. Enter the name of the .xsaccess file and choose Finish.
4. Add the following content to the .xsaccess file:

   ```json
   {
       "exposed" : true,
       "authentication" :
       {
           "method" : "Basic"
       }
   }
   ```
5. Commit and activate the .xsaccess file in the SAP HANA repository.
   To commit the .xsaccess file, right-click it and choose Team → Commit.
   To activate the .xsaccess file, right-click it and choose Team → Activate.

6.1.7 Creating Source Files (HTML & CSS & JS)

1. In the SAP HANA studio, open the SAP HANA Development perspective.
2. In the Project Explorer view, right-click your project, and choose New → Other.
   - Create an index.html for displaying data.
   - Create some css files for rendering html.
   - Create a JavaScript file for the main logic and for retrieving data from the SAP HANA server.
   - Add any other files you may need in your project, for example, images, JQuery library, and so on.
     For more information, refer to the atp sample source code.
3. Commit and activate the source files in the SAP HANA repository.
To commit the files, select all source files, right-click, and choose Team → Commit.

To activate the files, select all source files, right-click, and choose Team → Activate.

6.1.8 Adding OData Service

1. In the SAP HANA studio, open the SAP HANA Development perspective.
2. In the Project Explorer view, right-click your project, and choose New → Folder.
3. In the Folder Name field, enter OData and choose Finish.
4. Repeat steps 2 and 3 to create a child folder SBODEMOUS under OData.
5. Right-click the SBODEMOUS folder, and choose New → File.
6. Create an OData service definitions file. For example, create file context.xsodata. You can refer to the context.xsodata file in the atp sample source code for the content.
7. Create views.
   You should create the following views for this sample:
   - interest.view
   - opin.view
   - opportunity.view
   - partner.view
   - product.view
   - sales.view
   - warehouse.view
   For more information, refer to the partner.view file in the atp sample source code for the content.
8. Replace all placeholders {COMPANY} with SBODEMOUS in all views and xsodata files.
9. Commit and activate the OData files in the SAP HANA repository.
   - To commit the files, right-click the OData folder, and choose Team → Commit.
   - To activate the files, right-click the OData folder, and choose Team → Activate.

6.1.9 Creating ATP Stored Procedures

You should create several stored procedures for ATP usage. For more information, refer to /atp/db/atp_free.sql in the atp sample source code, and import the stored procedures into your HANA database.

6.1.10 Creating User-Defined Stored Procedures

After you create the ATP stored procedures, you should create an entry in the Query Manager window of SAP Business One.
1. From the Tools menu of SAP Business One, choose Queries → Query Manager.
The Query Manager window appears.
2. Choose the New Stored Procedure button.
The User Defined Stored Procedure window appears.
3. Define the UDSP alias as atp for the stored procedure ATP_XAPP.
4. Choose the Save button.
The Save Query window appears.
5. Save the query in the General category and choose OK.

Note
To manage the authorization groups for the categories, choose the Manage Categories button.
For more information, see the Create/Edit Categories Window section in SAP Business One online help.
To manage the user authorization, from the SAP Business One Main Menu, choose Administration → System Initialization → Authorizations.
For more information, see the Authorizations section in SAP Business One online help.

6.1.11 Testing the App

You can use either a web browser, or the browser widget of SAP Business One client to test the app.

• To use the web browser, perform the following steps:
  1. Open your web browser and navigate to
  2. Fill in all the fields and choose Login.
     The application logs in both HANA XS and SAP Business One, and redirect to the site you desire (in this case, is https://10.58.1.134:4300/sap/sbo/atp/)

• To use the browser widget of SAP Business One client, perform the following steps:
  1. Log on to the SAP Business One client.
  2. Enable the cockpit and log on again.
  3. Open a browser widget in My Cockpit, and in the browser settings, enter the URL https://hana_server_IP:43<SID>/sap/sbo/atp/ to test the app.
6.2 Packaging App ATP

6.2.1 Exporting Delivery Unit

1. In the SAP HANA studio, open the SAP HANA Modeler perspective.
2. Select your SAP HANA server.
3. Create a delivery unit for your app. For example, here we can use the name ATP for the delivery unit.
4. From the File menu, select Export...
5. Choose SAP HANA Content → Delivery Unit and choose Next.
6. Select your SAP HANA server and choose Next.
7. Select the delivery unit to export. Here we choose ATP.
8. Select a location for the exported delivery unit. Here we choose Export to Client.
9. To confirm the settings, choose Next.
10. To start the export operation, choose Finish.

6.2.2 Using SAP Business One DI API to Package Master Data of UDQ and UDSP

You should develop an add-on using SAP Business One DI API to package the UDQ and UDSP master data. You can get the sample source code from DI_OUQR.cs in the zipped package.
6.2.3 Creating Extension Registration Files

1. From the SAP Business One SDK Tools folder, run ExtensionRegDataGen.exe. The ExtensionRegDataGen.exe file is typically located at: "..\Program Files (x86)\sap\SAP Business One SDK\Tools\ExtensionRegDataGen".
2. Specify the partner information and the app information.
3. Choose Export to generate an ARD file. Here we save the file with the name atp.ard.

6.3 Deploying App ATPs

6.3.1 Importing Delivery Units

1. In the SAP HANA studio, open the SAP HANA Modeler perspective.
2. From the File menu, choose Import....
3. Choose SAP HANA Content → Delivery Unit and choose Next.
4. Select your SAP HANA server and choose Next.
5. Select the delivery unit to import. Here you import from the client: browse to the location and choose the delivery unit you exported.
6. Choose Finish to start the export operation.

6.3.2 Registering Apps in SAP Business One

1. Start the SAP Business One client and log on with a super user or a permitted user for App Center.
2. From the Tools menu, choose App Center.
3. In the top left-hand corner of the App Center window, choose the Register New App button.
4. In the pop-up window, select the ARD file you exported, and choose OK.
6.3.3 Assigning Apps to Your Company

1. Start the SAP Business One client and log on with a super user or a permitted user for App Center.
2. From the Tools menu, choose App Center.
3. Drag your app from the Unassigned Apps area to the Assigned Apps area.
   Your App appears in the Assigned Apps area.

6.3.4 Assigning User Permissions

1. Start the SAP Business One client and log on with a super user or a permitted user for App Center.
2. From the Tools menu, choose App Center.
3. In the top left-hand corner of the App Center window, choose the User Authorization button.
   The user list of the current company appears on the right side of the App Center window.
4. Select the user(s) to whom you want to assign permission.
   Note
   Super users have permission by default.
6.3.5  Activating OData Service for the Current Company

In the sample delivery unit, you exported OData definition files with a dedicated name SBODEMOUS. If you want to activate the service for a company other than SBODEMOUS, you should follow the steps below to create a copy of the OData definition files and activate them:

1. Under the OData folder, create another child folder and name it according to the company name.
2. Copy all the OData definition files (view and xsodata) to the folder.
3. Replace all placeholders (COMPANY) with the company name in all views and xsodata files.
4. Save, commit, and activate the OData files.

6.3.6  Running Your DI API Add-on

To import your stored procedure, application initialization data, or app level tables, run the DI API add-on that was created in section 5.2.2.2.

6.3.7  Running Your Apps

1. Log on to the SAP Business One client.
2. In the Sales Opportunity window, add Interest Range for the sales opportunity.
3. Open a browser widget in My Cockpit, and in the browser settings, enter the URL https://hana_server_IP:43<SID>/sap/sbo/atp/to run the app.

You can view the amount of the product which is of interest as an opportunity to your business partner.

---

**Step 1: Select a BP or Sales Employee**

<table>
<thead>
<tr>
<th>Opportunity</th>
<th>BP Code</th>
<th>BP Name</th>
<th>Sales Employee</th>
<th>Status</th>
<th>Potential Amount</th>
<th>Weighted Amount</th>
<th>Gross Profit Total</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C20000</td>
<td>Norm Thompson</td>
<td>O</td>
<td>29003</td>
<td>17403</td>
<td>26000</td>
<td></td>
<td>View, Copy To Sales Order</td>
</tr>
<tr>
<td>16</td>
<td>C20000</td>
<td>Norm Thompson</td>
<td>O</td>
<td>10000</td>
<td>600</td>
<td>6000</td>
<td></td>
<td>View, Copy To Sales Order</td>
</tr>
<tr>
<td>25</td>
<td>C20000</td>
<td>Norm Thompson</td>
<td>O</td>
<td>29003</td>
<td>17403</td>
<td>26000</td>
<td></td>
<td>View, Copy To Sales Order</td>
</tr>
<tr>
<td>37</td>
<td>C20000</td>
<td>Norm Thompson</td>
<td>O</td>
<td>31000</td>
<td>18000</td>
<td>12000</td>
<td></td>
<td>View, Copy To Sales Order</td>
</tr>
<tr>
<td>41</td>
<td>C20000</td>
<td>Norm Thompson</td>
<td>L</td>
<td>10000</td>
<td>2000</td>
<td>4000</td>
<td></td>
<td>View, Copy To Sales Order</td>
</tr>
<tr>
<td>52</td>
<td>C20000</td>
<td>Norm Thompson</td>
<td>O</td>
<td>10000</td>
<td>6000</td>
<td>4000</td>
<td></td>
<td>View, Copy To Sales Order</td>
</tr>
<tr>
<td>64</td>
<td>C20000</td>
<td>Norm Thompson</td>
<td>O</td>
<td>25000</td>
<td>5000</td>
<td>10000</td>
<td></td>
<td>View, Copy To Sales Order</td>
</tr>
</tbody>
</table>

**Step 2: Select a product of Interest**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Products of Interest</td>
<td>Intel P4 2.4 GHz</td>
</tr>
</tbody>
</table>

**Step 3: Product amount is displayed**

<table>
<thead>
<tr>
<th>Warehouse</th>
<th>General Warehouse</th>
<th>Amount after 3 months</th>
<th>Amount after 6 months</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Amount</td>
<td>1018</td>
<td>1018</td>
<td>1018</td>
</tr>
<tr>
<td>West Cost Warehouse</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Dropship Warehouse</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Consignment Warehouse</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Product: C90003