Developer Guide
App Framework for SAP Business One, version for SAP HANA
SAP Business One 9.1, version for SAP HANA
SAP HANA Platform Edition 1.0 SPS 07 Rev74
Document Version: 1.1 – 2014-09-15

PUBLIC

Working with App Framework for SAP Business One, version for SAP HANA

Typographic Conventions

Type Style	Description
Example	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.
Example	Emphasized words or expressions.
`EXAMPLE`	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.
`Example`	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.
`Example`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE`	Keys on the keyboard, for example, `F2` or `ENTER`.

Document History

Version	Date	Change
1.0	2014-06-27	First version for SAP Business One 9.1, version for SAP HANA
1.1	2014-09-15	Naming change of extreme app and extreme app framework

Table of Contents

1 Introduction

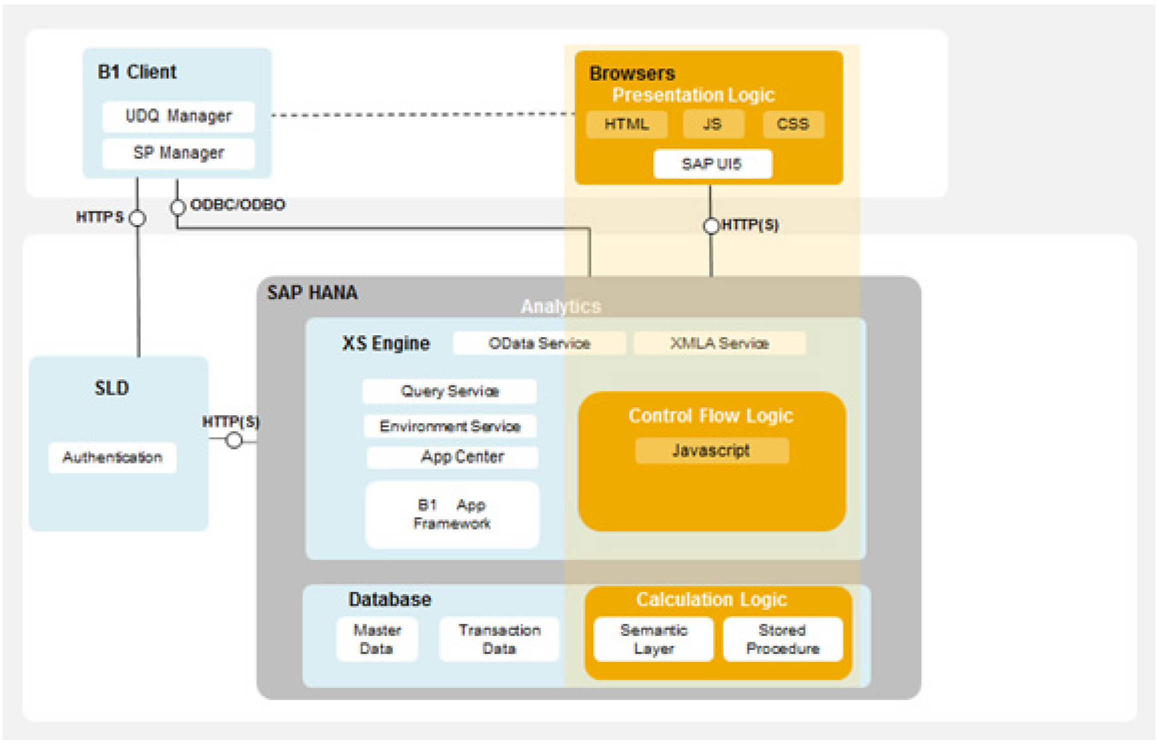
This document describes how to work with the App Framework for SAP Business One, version for SAP HANA. To do so, you should have a basic familiarity with software development, web services, and SAP HANA.

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**: 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 *SAP Business One Extension Manager* to import apps, assign/unassign companies.
  - You can manage user authentication in the *Authorizations* window of the SAP Business One client.
- **Single Sign On (SSO) in SAP Business One**

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

Terms and Definitions

The following terms are used in this document.

Term	Definition
SAP HANA	SAP High-Performance Analytical Appliance
SAP HANA XS	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.
App available with SAP Business One, version for SAP HANA	A lightweight application with real time analytics capability.
DI API	SAP Business One SDK Data Interface Application Programming Interface

Related Documentation

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

Document	Location
SAP HANA Installation Guides	http://help.sap.com/hana_appliance
SAP HANA Developer Guide	http://help.sap.com/hana/hana_dev_en.pdf
SAP Business One Administrator's Guide, version for SAP HANA	http://service.sap.com/smb/sbocustomer/documentation The document includes the installation information of the App Framework for SAP Business One, version for SAP HANA.
SDK online help file SDK_EN.chm	http://service.sap.com/smb/sbocustomer/documentation 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
SAP Business One Online Help	http://help.sap.com/businessone900

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 the version for SAP HANA.

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.

  For more information about installing SAP HANA server, HANA client, and HANA studio, see SAP HANA *Installation and Upgrade Information* on SAP Help Portal at http://help.sap.com/hana_appliance.
- 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 App artifacts, so you need to install it before implementing the lifecycle management of App. The `ExtensionPackage.exe` inside SDK Tools is used for packaging your App.

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.

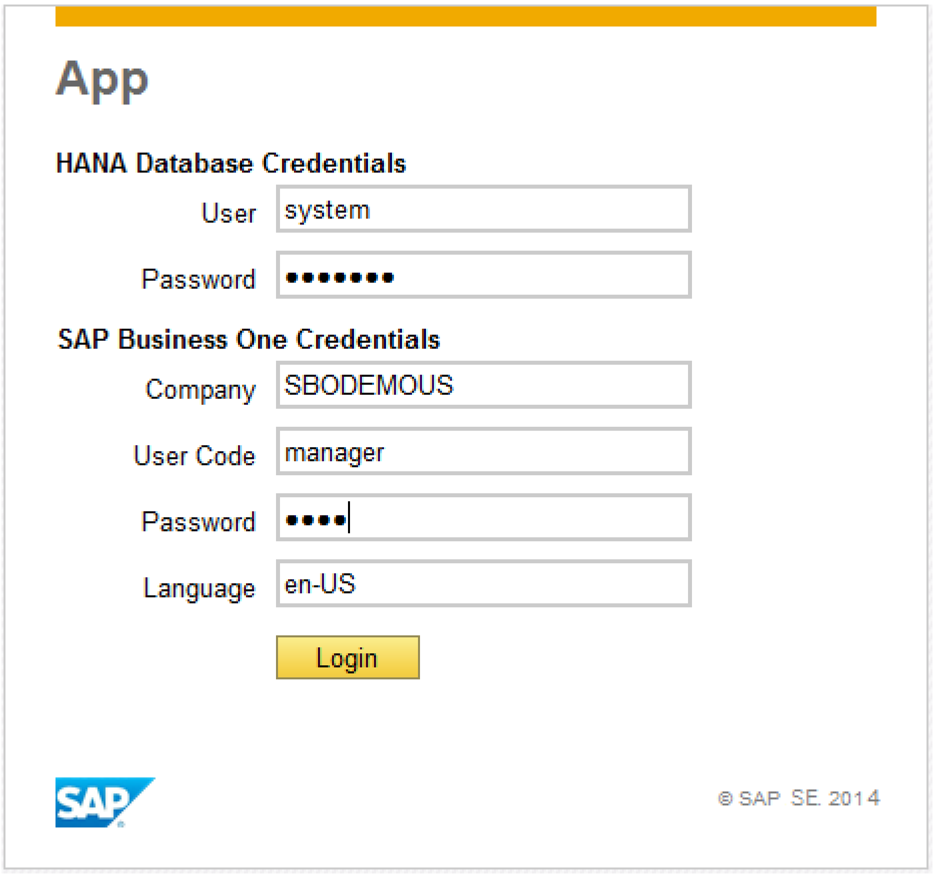
Procedure

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:

```
<!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);
                    },
                    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="language" 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.


App

HANA Database Credentials

User	system
Password	●●●●●●●●

SAP Business One Credentials

Company	SBODEMOUS
User Code	manager
Password	●●●●
Language	en-US

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:

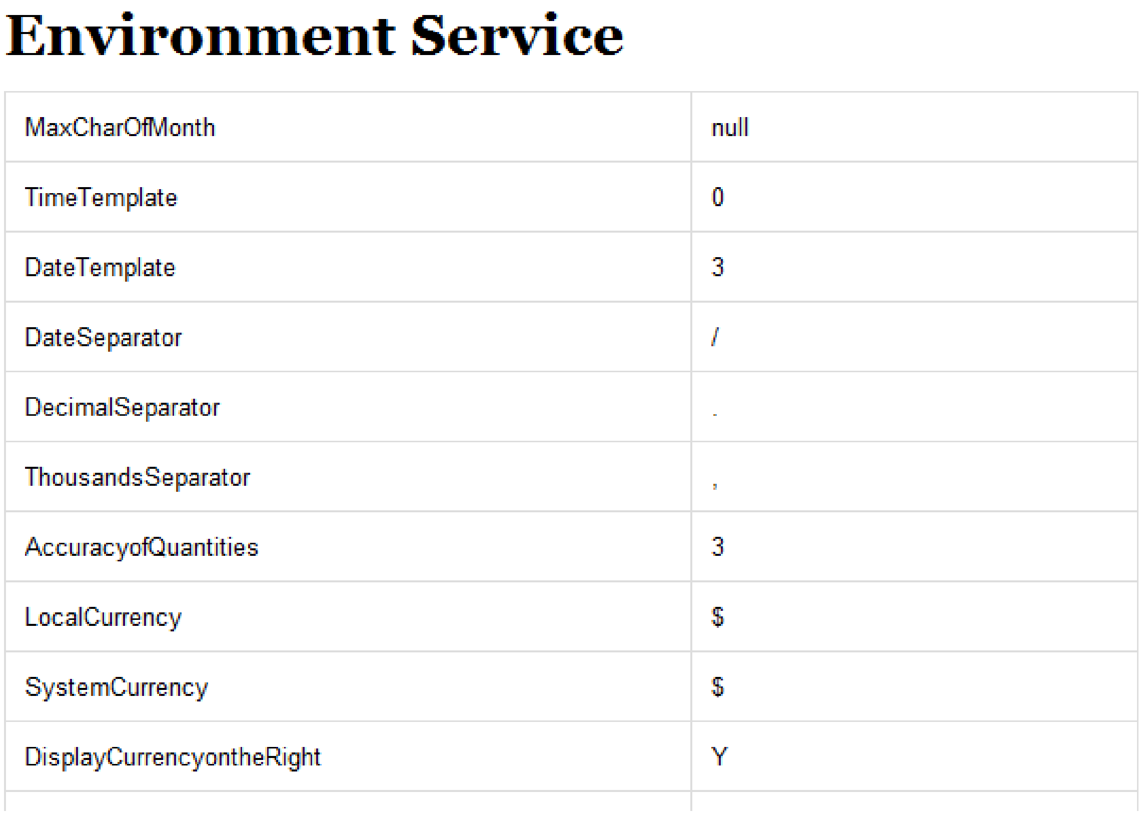
```
<!DOCTYPE html>
<html>
<head>
    <script src="jquery.js"></script>
    <script>
        $(function () {
            /*Environment service sample code*/
            $("#env").click(function () {
                var button = $(this);
                button.attr("disabled", "disabled");
                $.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>
    </body>
    </html>
```

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

MaxCharOfMonth	null
TimeTemplate	0
DateTemplate	3
DateSeparator	/
DecimalSeparator	.
ThousandsSeparator	,
AccuracyofQuantities	3
LocalCurrency	$
SystemCurrency	$
DisplayCurrencyontheRight	Y

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

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

- The Login page: https://hana_server_IP:43<SID>/sap/sbo/portal/?site=/your/company/myapp/ (for example, https://10.58.1.134:4300/sap/sbo/portal/?site=/your/company/myapp/).

  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/)
- The demo page: https://hana_server_IP:43<SID>/sap/sbo/demo/.

  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)
              "U1lTVEVNOm1hbmFnZXI=", (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)
}
```

Property	Description
Type	Indicates whether the query service executes a UDQ(sql) or a UDSP(sp).
Category	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`.
Name	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`.
Param	A JavaScript array of input, output, and inout parameters for executing the specific UDQ/UDSP. • Date: Data is double quote embraced and is in the format of "YYYY-MM-DD HH:mm:ss.SSS". • Integer: Data is either double quote embraced or not. • Blob: Data is double quote embraced and is base64 encoded. • String and other data types: Data is double quote embraced. • Null: Represents null/empty data for any kinds of type.

Format	Indicates the output data format that the query service is going to apply. This parameter is optional. The default value is *CondenseJSON*,
	The *CondenseJSON* format generates JSON data as concisely as possible. The following code shows the difference between *CondenseJSON* and *JSON* formats:
	• *CondenseJSON*
	`{`
	`        "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"`

Property	Description

```
        }]],
        "data": [[["General Warehouse",
        980,
        980,
        980],
        ["West Cost Warehouse",
        0,
        0,
        0],
        ["Dropship Warehouse",
        0,
        0,
        0],
        ["Consignmentl 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",
```

Property	Description

	```
	                  "type": "INTEGER"
	            }]],
	            "data": [[{
	                    "WAREHOUSE": "General Warehouse",
	                    "QTY1": 980,
	                    "QTY2": 980,
	                    "QTY3": 980
	            },
	            {
	                    "WAREHOUSE": "West Cost Warehouse",
	                    "QTY1": 0,
	                    "QTY2": 0,
	                    "QTY3": 0
	            },
	            {
	                    "WAREHOUSE": "Dropship Warehouse",
	                    "QTY1": 0,
	                    "QTY2": 0,
	                    "QTY3": 0
	            },
	            {
	                    "WAREHOUSE": "Consignmentl Warehouse",
	                    "QTY1": 0,
	                    "QTY2": 0,
	                    "QTY3": 0
	            }]],
	            "param": []
	    }
	```
	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],
    "data": [data 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`.

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:

Response Fields	DB Fields	DB Table
`MaxCharOfMonth`	`CharMonth`	`OADM`
`TimeTemplate`	`TimeFormat`	`OADM`

Response Fields	DB Fields	DB Table
DateTemplate	DateFormat	OADM
DateSeparator	DateSep	OADM
DecimalSeparator	DecSep	OADM
ThousandsSeparator	ThousSep	OADM
AccuracyofQuantities	QtyDec	OADM
LocalCurrency	MainCurncy	OADM
SystemCurrency	SysCurrncy	OADM
DisplayCurrencyontheRight	CurOnRight	OADM
PriceAccuracy	PriceDec	OADM
QueryAccuracy	QueryDec	OADM
PercentageAccuracy	PercentDec	OADM
TotalsAccuracy	SumDec	OADM
RateAccuracy	RateDec	OADM
MeasuringAccuracy	MeasureDec	OADM
CompanyName	CompnyName	OADM
Country	Country	OADM
State	State	OADM
Language	UserPrefs	OUSR

REQUEST

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

SUCCESSFUL RESPONSE

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

{
    "MaxCharOfMonth": "...",
    "TimeTemplate": "...",
    "DateTemplate": "...",
    "DateSeparator": "...",
    "DecimalSeparator": "...",
    "ThousandsSeparator": "...",
    "AccuracyofQuantities": "...",
    "LocalCurrency": "...",
    "SystemCurrency": "...",
    "DisplayCurrencyontheRight": "...",
    "PriceAccuracy": "...",
    "QueryAccuracy": "...",
    "PercentageAccuracy": "...",
    "TotalsAccuracy": "...",
    "RateAccuracy": "...",
    "MeasuringAccuracy": "...",
    "CompanyName": "...",
    "Country": "...",
    "State": "..."
}
```

UNSUCCESSFUL RESPONSE

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

{
    "error": "reason of failure"
}
```

3.1.1.4 Extension Assignment API

The Extension Assignment API maintains the mapping relationship between the extension and the assigned company. The frond-end and the server-side API are provided for you to consume:

```
/********************************************************************************
 *
 * Sample code of detecting whether user can access current package in front-end
 *
 *******************************************************************************/
$(function() {
    $.ajax({
        url: "/sap/sbo/platform/xapp",
        success: function(data) {
            data.results.any = any;
            var targetUri = "/sap/your-package/your-app/your-resource/",
                permitted = data.results.any(function(it) {
                    var pkg = '/' + it.package.replace(/\./gm, '/');
                    return (uri.indexOf(pkg) !== -1);
                });
            if (permitted) {
                // TODO: user is allowed to access current package, continue with the
action...
            } else {
                // TODO: user is NOT allowed to access current package, throw
exception will do
            }
        },
        error: function(xhr, status, error) {
            console.log("failed to retrieve extension assignment info");
        }
    });

    function any() {
        if (typeof arguments[0] === "function") {
            var predicate = arguments[0],
                match = false;
            this.forEach(function(it) {
                if (!match) {
                    match = predicate(it);
                }
            });
            return match;
        } else {
            return this.length > 0;
        }
    }
});

/********************************************************************************
 *
 * Sample code of detecting whether user can access current package in server-side
 *
 ********************************************************************************/
```

```
$.root = "sap.sbo.cockpit.bc";
$.import("sap.sbo.cockpit.bc.common", "core");
var Assignment = $.require("util.assignment").Assignment;
var permitted = new Assignment().permitted();
if (permitted) {
    // TODO: user is allowed to access current package, continue with the action...
} else {
    // TODO: user is NOT allowed to access current package, throw exception will do
}
```

You must perform the additional steps to enable HTTP connector for SLD. Do the following:

1. Open `/opt/sap/SAPBusinessOne/Common/tomcat/conf/web.xml`, find the following XML element and delete it:

```
<security-constraint>
                <display-name>Security Constraint</display-name>
                <web-resource-collection>
                        <web-resource-name>Protected Area</web-resource-name>
                        <url-pattern>/*</url-pattern>
                </web-resource-collection>
                <user-data-constraint>
                        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
                </user-data-constraint>
        </security-constraint>
```

2. Open `/opt/sap/SAPBusinessOne/Common/tomcat/conf/server.xml`, add the following XML element

```
<Connector port="40015" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="40000" />
```

3. Restart tomcat via `/etc/init.d/sapb1servertools restart`.

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.

Note

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

```
<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

```
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

```
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:

```
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',
ConfirmationDocumnets = <any>'OWDD',
ConfirmationLevel = <any>'OWST',
ConfirmationTemplates = <any>'OWTM',
ContactWithCustAndVend = <any>'OCLG',
ContractTemplete = <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>'OSPP',
    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

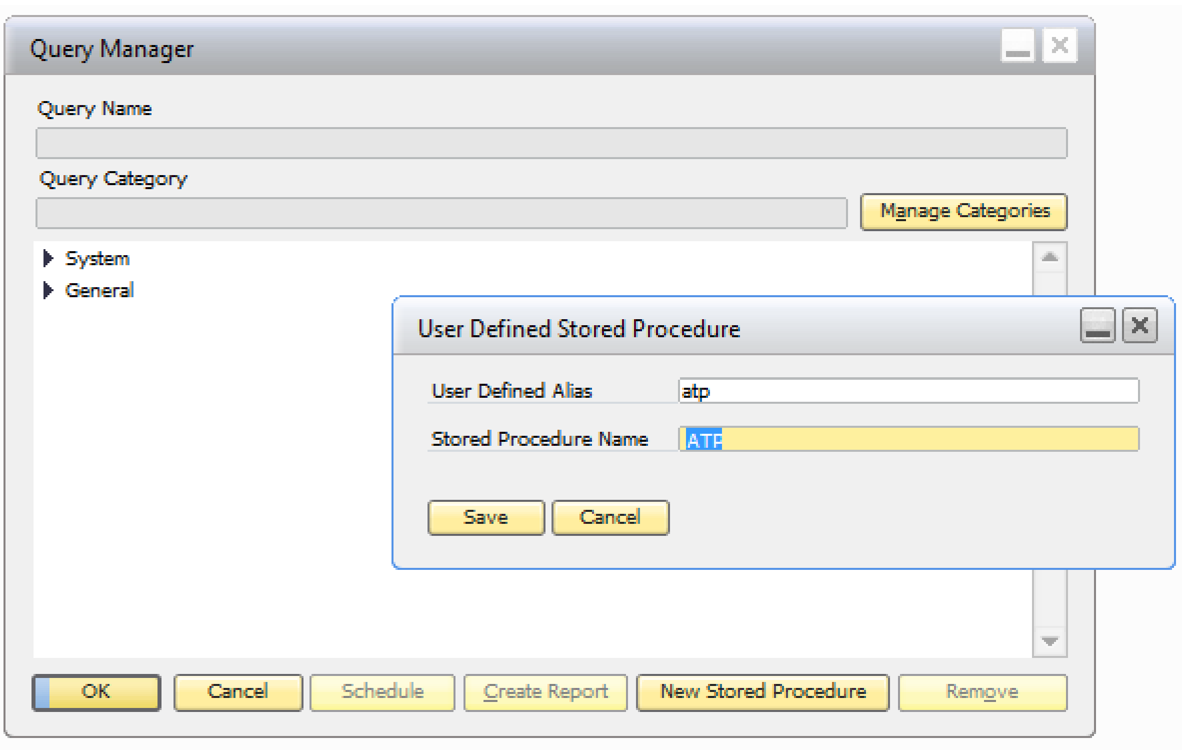
You can define and deploy your own stored procedures and maintain the master data of the procedures.

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 Alias: atp

Stored Procedure Name: ATP

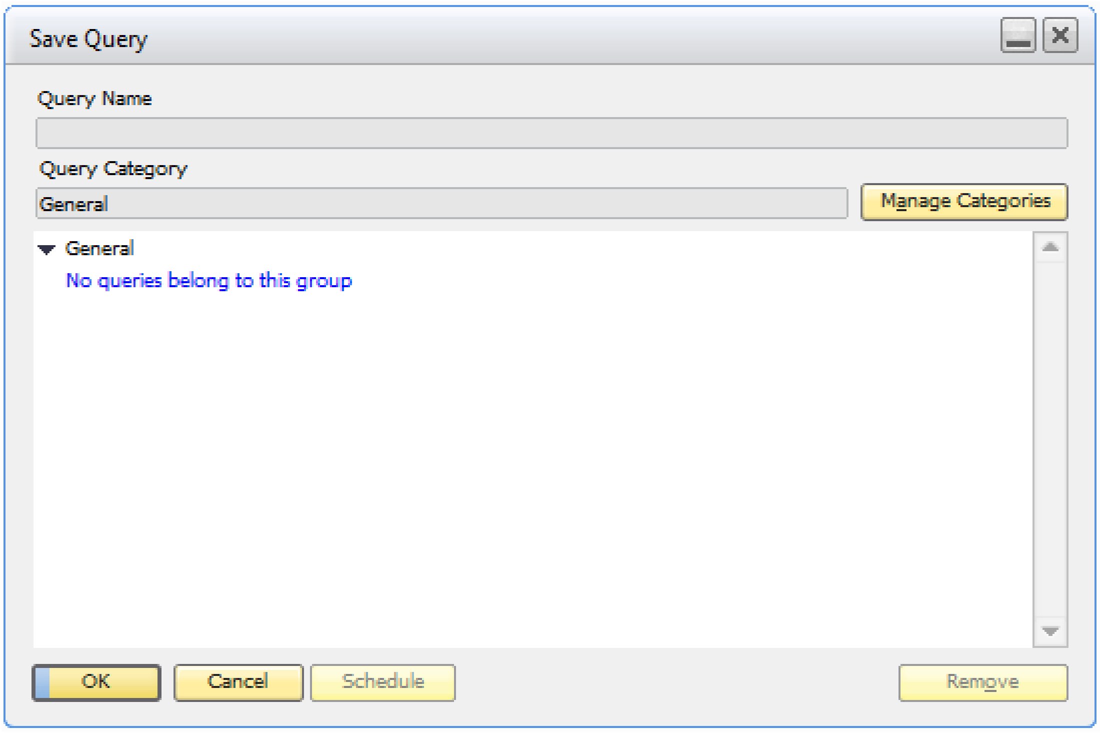
Note

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.

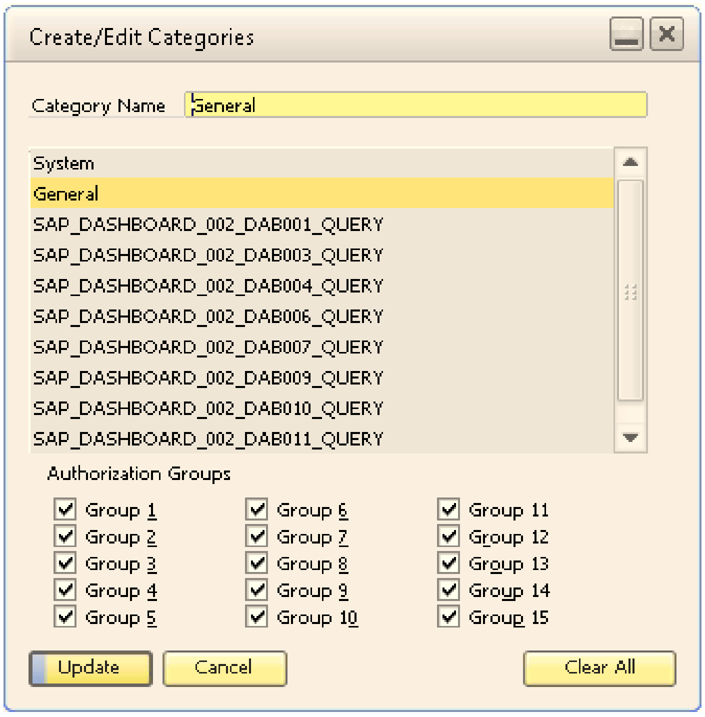


5. Save the query in the *General* category and choose *OK*.

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.


Create/Edit Categories

Category Name: General

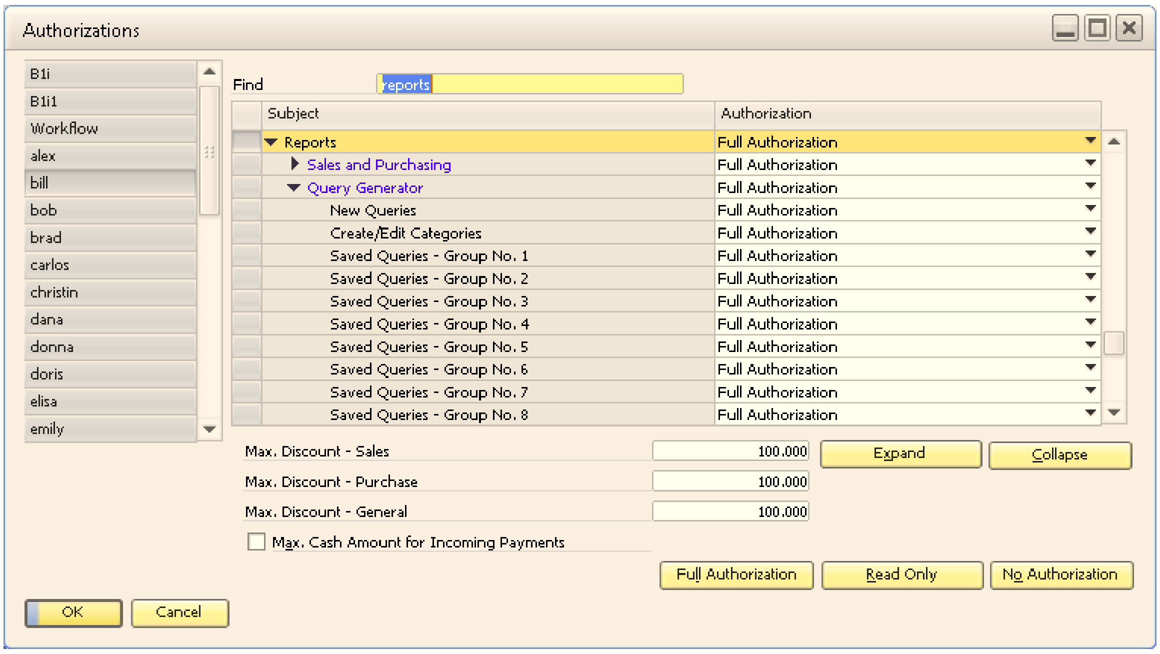
System
General
SAP_DASHBOARD_002_DAB001_QUERY
SAP_DASHBOARD_002_DAB003_QUERY
SAP_DASHBOARD_002_DAB004_QUERY
SAP_DASHBOARD_002_DAB006_QUERY
SAP_DASHBOARD_002_DAB007_QUERY
SAP_DASHBOARD_002_DAB009_QUERY
SAP_DASHBOARD_002_DAB010_QUERY
SAP_DASHBOARD_002_DAB011_QUERY

Authorization Groups

✓ Group 1	✓ Group 6	✓ Group 11
✓ Group 2	✓ Group 7	✓ Group 12
✓ Group 3	✓ Group 8	✓ Group 13
✓ Group 4	✓ Group 9	✓ Group 14
✓ Group 5	✓ Group 10	✓ Group 15

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.



Subject	Authorization
Reports	Full Authorization
Sales and Purchasing	Full Authorization
Query Generator	Full Authorization
New Queries	Full Authorization
Create/Edit Categories	Full Authorization
Saved Queries - Group No. 1	Full Authorization
Saved Queries - Group No. 2	Full Authorization
Saved Queries - Group No. 3	Full Authorization
Saved Queries - Group No. 4	Full Authorization
Saved Queries - Group No. 5	Full Authorization
Saved Queries - Group No. 6	Full Authorization
Saved Queries - Group No. 7	Full Authorization
Saved Queries - Group No. 8	Full Authorization

Max. Discount - Sales 100.000
Max. Discount - Purchase 100.000
Max. Discount - General 100.000
☐ Max. Cash Amount for Incoming Payments

4 User Interface Guidelines

The user interface guidelines provide best practices for designing the user interface of your SAP Business One Apps. 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 Apps

5.1.1 Packaging App Artifacts in SAP 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.

For more information, refer to the section 6.2.1 in the tutorial.

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

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.oUserQuerie
s);
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 Packaging Extension Data Files

The Extension Package tool is a component in SAP Business One Software Development Kit. To install the Extension Package tool, you should install SAP Business One Software Development Kit, and select *Tools → ExtensionPackage*.

Using the Extension Package tool, you are able to do the following:

- Package your extension for lightweight deployment
- Create an add-on registration data (ARD) file
- Import and edit an existing ARD file

> Note
>
> You can package your extension together with your app, or you can package your app only.

To package the extension files to a zip file and create a new file containing add-on registration data, do the following:

1. In …`\SAP\SAP Business One SDK\Tools\ExtensionPackage`, run the `ExtensionPackage.exe` file.
2. In the *Extension Registration Data Generator* window, expand *Basic Information*, and specify the following fields:
   - o *Extension Name* – Enter the name of the extension. This field is mandatory.
   - o *Extension Version* – Enter the version of the extension for which you want to package and generate the ARD file. This field is mandatory.
   - o *Extension Provider* – Enter the name of the SAP partner that creates and owns the extension. For example, the name of your company.
   - o *Extension Namespace* – Enter a name for the folder in which SAP Business One places the extension after a user registers the extension in the application.
   - o *Supported Database* – Specify the database in which the extension works. This field is mandatory.
   - o *Contact Data.* – Enter contact information for the SAP partner that creates and owns the extension. For example, enter the URL of your company's Website.
3. Expand *Extension File*, from the *SBO App on SAP HANA* tab, specify the information of your app:

- *Choose Delivery Unit* – Choose *Open* to import the delivery unit you have created in section 5.1.1.
- *App Name* – Specify the name of your app for the version for SAP HANA.
- *Package Name.* – Specify the package name of your app for the version for SAP HANA.

  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.

If you also want to package your extension for lightweight deployment, from the *SBO 32-Bit Client Add-on* tab or the *SBO 64-Bit Client Add-on* tab, select the path of the executable file of your 32-bit or 64-bit add-on, and select the files that should be packaged in the zip file.

4. (Optional) Expand *Deployment Steps*, and select the COM dlls to register for your 32-bit or 64-bit add-on.
5. (Optional) Expend *SBO Compatibility*; you can specify the versions of SAP Business One with which the add-on is compatible.

   In the *Compatible with SAP Business One* area, specify the following:

   - *Version From* – Enter the earliest version of SAP Business One with which the extension is compatible.
   - *To* – Enter the latest version of SAP Business One with which the lightweight add-on is compatible.

     Example

     Enter compatible versions of SAP Business One with the format **`<xxx.yyy.xx>`**, for example, **`910.000.00.`**
6. (Optional) Expand *Parameters*; you can optionally specify shared parameters and parameters for the extension.
   - Shared Parameters - The parameters or configuration required to run the extension. The parameters are shared in all extension instances running on different companies.
   - Parameters - The parameters or configurations required to run the extension.
7. Choose the *Package* button.
8. In the *Save As* window, specify the location where you want to save the file and choose the *Save* button.

   Note

   If you just want to create a new file containing add-on registration data, choose the *Export* button, and save the ARD file.

   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.

For more information, see *How to Package and Deploy SAP Business One Extensions for Lightweight Deployment*.

5.2 Deploying Apps

To deploy an app, perform the following steps:

1. Import your app zip file to SAP Business One Extension Manager.
2. Deploy the app to companies.
3. Run the app in SAP Business One.

5.2.1 Importing Your Extension to SAP Business One Extension Manager

Caution

Access to SAP Business One Extension Manager is under SAP Business One user authorization. Only a site user can open it. For more information about site users, see SAP Business One online help.

To access SAP Business One Extension Manager, do the following:

1. In the SAP Business One client, from the *Main Menu*, choose *Administration → Add-Ons → Add-On Administration*.

   The *Add-On Administration* window appears.
2. In the *Add-On Administration* window, click the *Manage Extensions for Lightweight Deployment* hyperlink.

   A Web browser opens and displays the logon page of System Landscape Directory (SLD).
3. To log on, enter the site user name and password and choose the *Log On* button.

   The *SAP Business One Extension Manager* window appears.

   Note

   Alternatively, you can access SAP Business One Extension Manager directly from a Web browser on the machine on which the System Landscape Directory (SLD) service is running using the following URL: https://<hostname>:<port>/ExtensionManager.

To import your extension to SAP Business One Extension Manager, perform the following steps:

1. In the *SAP Business One Extension Manager* window, on the *Extensions* tab, choose the *Import* button.

   The *Extension Import Wizard* window appears.
2. Choose the *Browse* button to locate the zip file of your extension (created in section 5.1.3), and choose *Upload*.

   The basic information of the extension appears.
3. Choose *Next* to optionally specify the value of the shared parameters.

   The shared parameters are the parameters or configuration required to run the extension. The parameters are shared in all extension instances running on different companies.

   The *Shared Parameters* table displays all shared parameters that are defined in the Extension Package tool when you package your extension.
4. Choose *Next*.

   On the *Finish* tab, we recommend that you continue to assign this extension to a company.

   After you click the *Finish import and run the company assignment wizard* hyperlink, the *Company Assignment Wizard* window appears.

5.2.2 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.

   Use one of the following two ways to assign an extension to a company:

   - Run the company assignment wizard
   - Run the extension assignment wizard

Running the Company Assignment Wizard

1. In the last step of the *Extension Import Wizard* window, choose the *Finish import and run the company assignment wizard* hyperlink.

   The *Company Assignment Wizard* window appears.

2. From the *Specify Company* tab, select a company to which you want to assign this extension, and choose *Next*.

3. Optionally, from the *Specify Parameters* tab, specify the value of the parameters and choose *Next*.

   The parameters are the parameters or configuration required to run the extension. The parameters are specific for the extension in this company.

   The *Parameters* table displays all parameters that are defined in the Extension Package tool when you package your extension.

4. From the *Specify Setup Mode* tab, select the default startup mode of this extension, specify the user preferences, and choose *Next*.

   - *Default Startup Mode*: The default startup mode determines how the extension is launched for all users that are connected to the company.
     - *Automatic* - SAP Business One starts the extension automatically. Users can stop automatically started add-ons with no impact on SAP Business One. A warning message informs users when the extension stops.
     - *Manual* - SAP Business One does not start the extension automatically. Users can start the extension at any time. A message informs users when a manually started extension is stopped.
     - *Mandatory* - SAP Business One starts the extension automatically. The extension is necessary for the successful operation of the SAP Business One application. The application launches the extension at start-up and shuts it down if the extension is terminated for any reason. Users cannot start or stop mandatory extensions.
     - *Disabled* - The extension is disabled.
   - *User Preferences*: All users are displayed in the *User Preferences* table, letting you set preferences for users in the company.
     - *Default* - User preferences for the extension come from the company preferences.
     - *Automatic* - SAP Business One starts the extension automatically. Users can stop automatically started extensions with no impact on SAP Business One.
     - *Manual* - SAP Business One does not start the extension automatically. Users can start the extension at any time.
     - *Disabled* - The extension is disabled for the selected user.

5. On the *Finish* tab, if you want to assign another extension, click the *Run the company assignment wizard again* hyperlink, and you can repeat the steps to assign the extension to another company.

6. Choose *Finish* to close the *Company Assignment Wizard* window.

7. To check the assigned extensions in a company, in the *SAP Business One Extension Manager* window, choose the *Company Assignment* tab. From *Company List*, select the company and check whether the extension is available.

Running the Extension Assignment Wizard

1. In the *SAP Business One Extension Manager* window, choose the *Company Assignment* tab.
2. From *Company List*, select the company to which you want to assign extensions.
3. In the *Extensions* area, choose *Assign*.

   The *Extension Assignment Wizard* window appears.
4. On the *Specify Extension* tab, select an extension that you want to assign to this company, and choose *Next*.
5. Optionally, on the *Specify Parameters* tab, specify the value of the parameters, and choose *Next*.

   The parameters are the parameters or configuration required to run the extension. The parameters are specific for the extension in this company.

   The *Parameters* table displays all parameters that are defined in the *Extension Registration Data Generator* tool when you package your extension.

   For more information, see *Specifying Parameters Information*.
6. From the *Specify Setup Mode* tab, select the default startup mode of this extension and specify the user preferences, and choose *Next*.
   - *Default Startup Mode*: The default startup mode determines how the extension is launched for all users that are connected to the company.
     - *Automatic* - SAP Business One starts the extension automatically. Users can stop automatically started add-ons with no impact on SAP Business One. A warning message informs users when the extension stops.
     - *Manual* - SAP Business One does not start the extension automatically. Users can start the extension at any time. A message informs users when a manually started extension is stopped.
     - *Mandatory* - SAP Business One starts the extension automatically. The extension is necessary for the successful operation of the SAP Business One application. The application launches the extension at start-up and shuts it down if the extension is terminated for any reason. Users cannot start or stop mandatory extensions.
     - *Disabled* - The extension is disabled.
   - *User Preferences*: All users are displayed in the *User Preferences* table, letting you set preferences for users in the company.
     - *Default* - User preferences for the extension come from the company preferences.
     - *Automatic* - SAP Business One starts the extension automatically. Users can stop automatically started extensions with no impact on SAP Business One.
     - *Manual* - SAP Business One does not start the extension automatically. Users can start the extension at any time.
     - *Disabled* - The extension is disabled for the selected user.
7. On the *Finish* tab, if you want to assign another extension, click the *Run the extension assignment wizard again* hyperlink, and repeat the steps to assign other extensions to this company.
8. Choose *Finish* to close the *Extension Assignment Wizard* window.

5.2.3 Running Apps

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

Note

Access to App 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.

5.2.3.1 Running Apps in Web Browser

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

1. Open your web browser and navigate to https://hana_server_IP:43<SID>/sap/sbo/portal/?site=/your/company/myapp/ (for example, https://10.58.1.134:4300/sap/sbo/portal/?site=/your/company/myapp/).
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/)

5.2.3.2 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.

5.2.3.3 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/xapp1";
```

6 Tutorial: Step by Step Building Apps: Building an App ATP

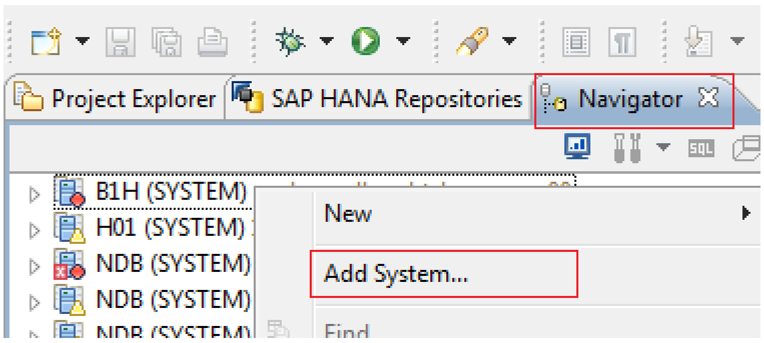
6.1 Developing an App ATP

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*.

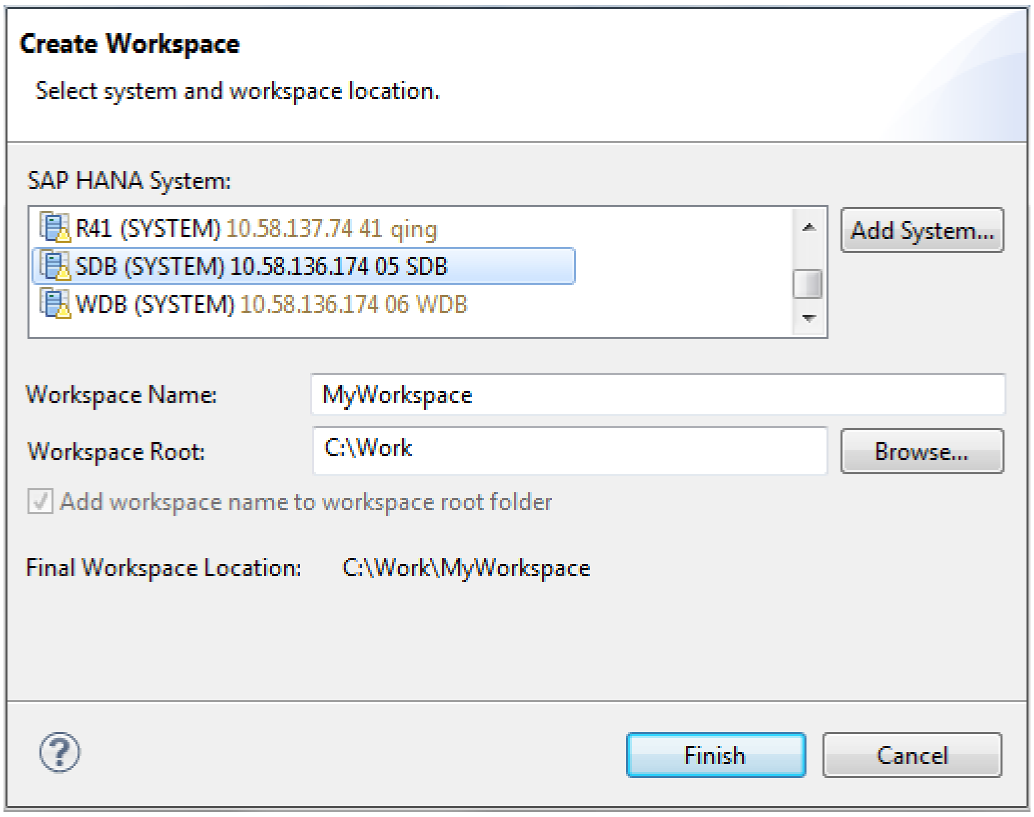
   The newly added system appears in the *Navigator* view.

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*:
   - Specify the SAP HANA system for which you want to create a new workspace.
   - Enter a workspace name, which can be anything you like, for example, the name of the SAP HANA system where the repository is located.
   - Specify where the workspace root directory should be located on your local file system, for example, `C:\users\username\workspaces`.


Create Workspace

Select system and workspace location.

SAP HANA System:

R41 (SYSTEM) 10.58.137.74 41 qing
SDB (SYSTEM) 10.58.136.174 05 SDB
WDB (SYSTEM) 10.58.136.174 06 WDB

Workspace Name: MyWorkspace

Workspace Root: C:\Work

☑ Add workspace name to workspace root folder

Final Workspace Location: C:\Work\MyWorkspace

   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.

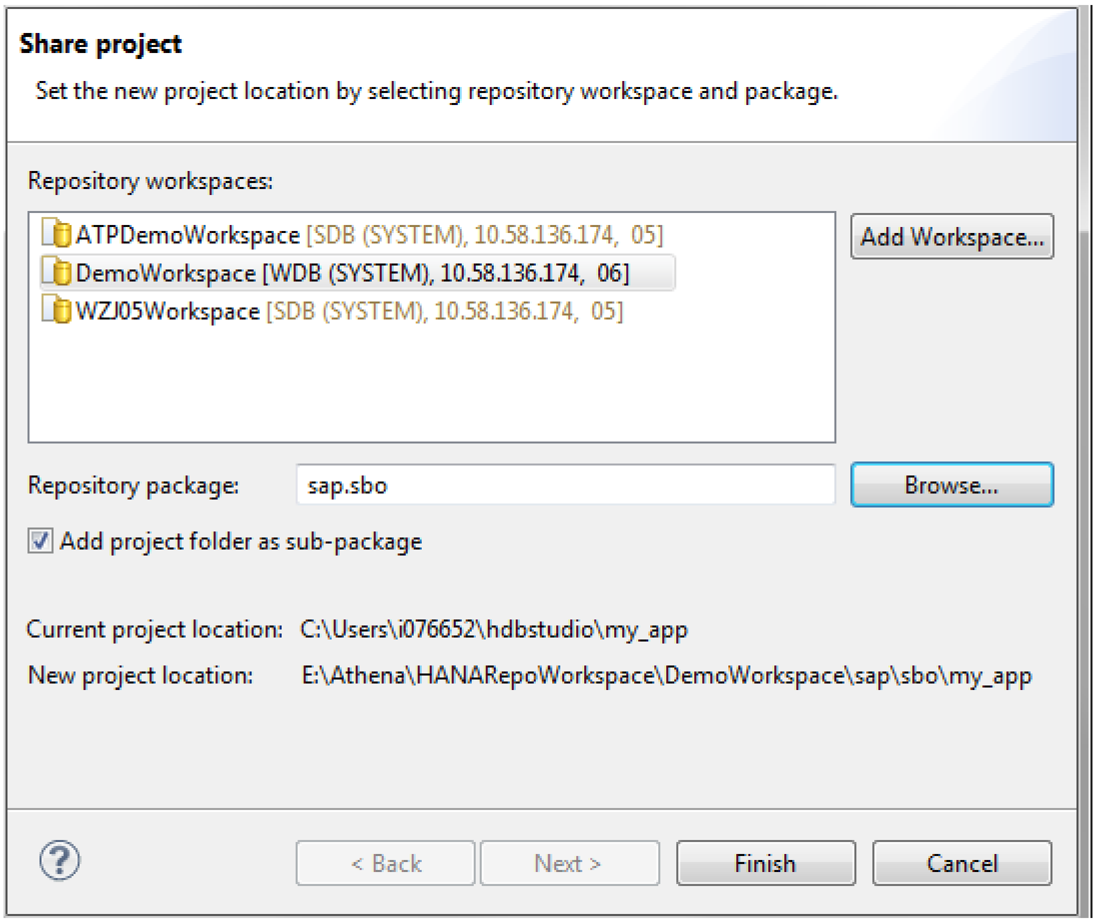
6. To create the new project, choose *Finish*.

   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.



Set the new project location by selecting repository workspace and package.

Repository workspaces:

ATPDemoWorkspace [SDB (SYSTEM), 10.58.136.174, 05]
DemoWorkspace [WDB (SYSTEM), 10.58.136.174, 06]
WZJ05Workspace [SDB (SYSTEM), 10.58.136.174, 05]

Repository package: sap.sbo

[✓] Add project folder as sub-package

Current project location: C:\Users\i076652\hdbstudio\my_app

New project location: E:\Athena\HANARepoWorkspace\DemoWorkspace\sap\sbo\my_app

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:

```
{
"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.xsdata` 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*.

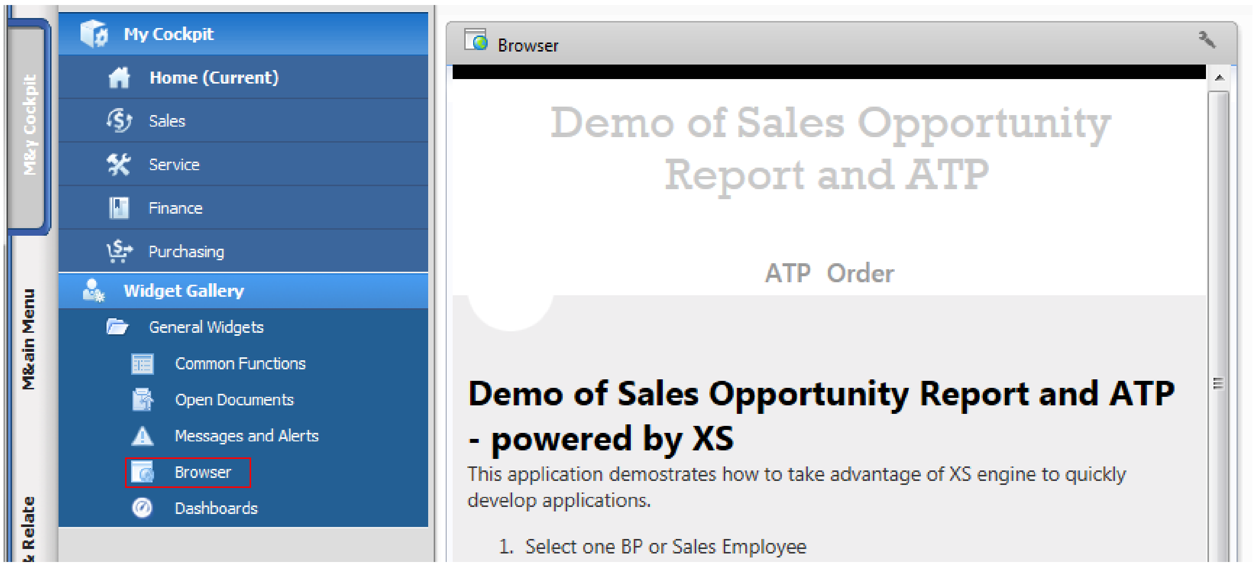
   For more information, see the *Authorizations* section in SAP Business One online help.

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 https://hana_server_IP:43<SID>/sap/sbo/portal/?site=/sap/sbo/atp/ (for example, https://10.58.1.134:4300/sap/sbo/portal/?site=/sap/sbo/atp/).
  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.



Demo of Sales Opportunity Report and ATP

Demo of Sales Opportunity Report and ATP - powered by XS

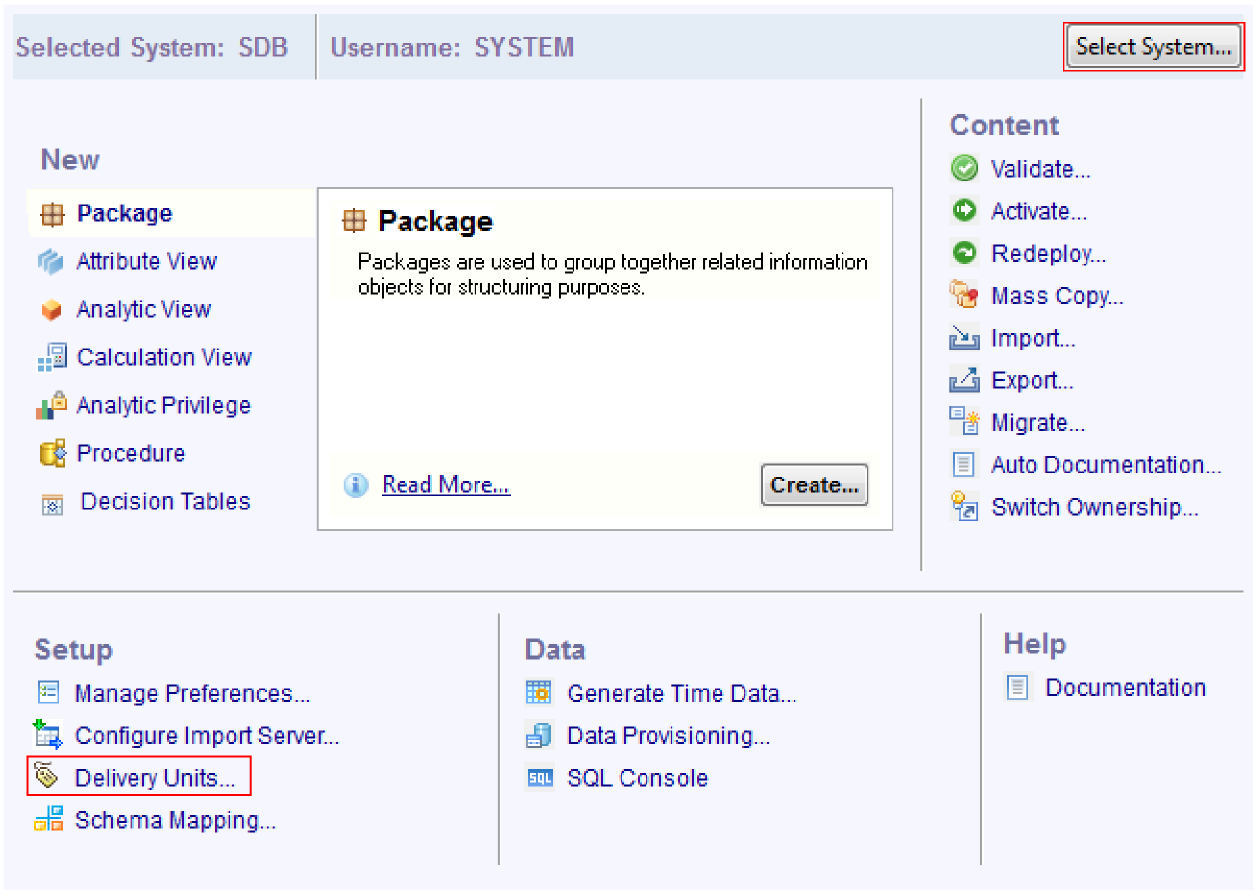
This application demostrates how to take advantage of XS engine to quickly develop applications.

1. Select one BP or Sales Employee

6.2 Packaging App ATP

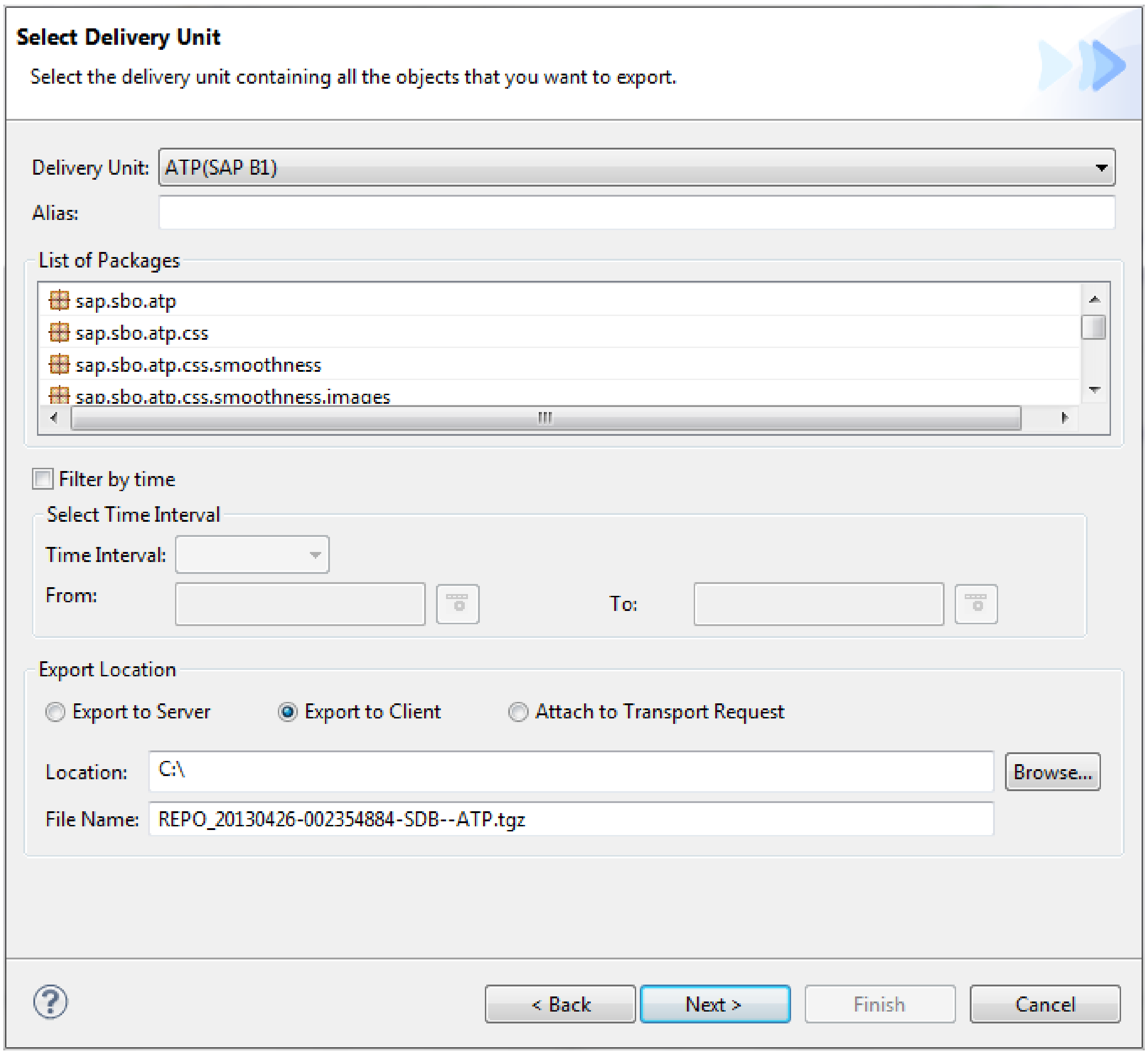
6.2.1 Exporting Delivery Unit

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.



Packages are used to group together related information objects for structuring purposes.

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*.


Select Delivery Unit

Select the delivery unit containing all the objects that you want to export.

Delivery Unit: ATP(SAP B1)

Alias:

List of Packages

- sap.sbo.atp
- sap.sbo.atp.css
- sap.sbo.atp.css.smoothness
- sap.sbo.atp.css.smoothness.images

☐ Filter by time

Select Time Interval

Time Interval:

From:

To:

Export Location

○ Export to Server [✓ ● Export to Client] ○ Attach to Transport Request

Location: C:\

File Name: REPO_20130426-002354884-SDB--ATP.tgz

9. To confirm the settings, choose *Next*.
10. To start the export operation, chose *Finish*.

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

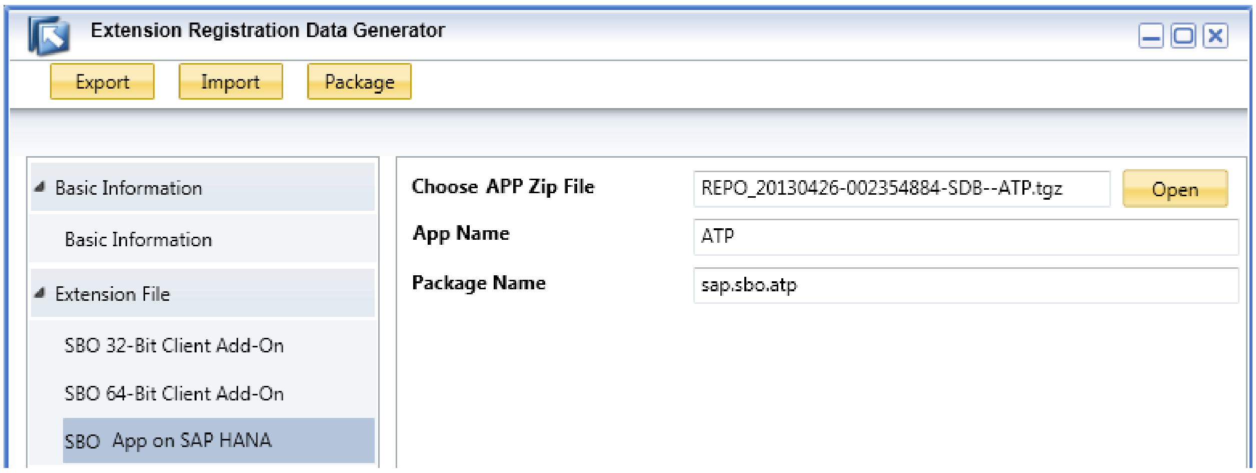
You should develop an add-on using SAP Business One DI API to package the UDQ and UDSP master data.

You can get the sample source code from `DI_OUQR.cs` in the zipped package.

6.2.3 Packaging App Data Files

1. From the SAP Business One SDK Tools folder, run `ExtensionPackage.exe`. The `ExtensionPackage.exe` file is typically located at: `..\Program Files (x86)\SAP\SAP Business One SDK\Tools\ExtensionPackage`.
2. On the *Basic Information* tab, specify the basic information of your extension.
3. On the *SBO App on SAP HANA* tab of the *Extension File* area, specify the information for your app.

   For more information, see *Packaging Extension Data Files*.
4. Click the *Package* button to save the files into a single zip file.



Choose APP Zip File: REPO_20130426-002354884-SDB--ATP.tgz
App Name: ATP
Package Name: sap.sbo.atp

6.3 Deploying App ATP

6.3.1 Importing Apps to SAP Business One Extension Manager

Caution

Access to SAP Business One Extension Manager is under SAP Business One user authorization. Only a site user can open it. For more information about site users, see SAP Business One online help.

1. Access SAP Business One Extension Manager directly from a Web browser on the machine on which the System Landscape Directory (SLD) service is running using the following URL: https://<hostname>:<port>/ExtensionManager.
2. In the *SAP Business One Extension Manager* window, on the *Extensions* tab, choose the *Import* button.

   The *Extension Import Wizard* window appears.
3. Choose the *Browse* button to locate the zip file of your extension, and choose *Upload*.
4. Choose *Next* to optionally specify the value of the shared parameters.
5. Choose *Next* to finish. On the *Finish* tab, we recommend that you continue to assign this extension to a company.

6.3.2 Assigning Apps to Your Company

1. After you imported the App to SAP Business One Extension Manager, you click the *Finish import and run the company assignment wizard* hyperlink, and the *Company Assignment Wizard* window appears.
2. From the *Specify Company* tab, select a company to which you want to assign this extension, and choose *Next*.
3. You can optionally specify the value of the parameters in the *Specify Parameters* tab and choose *Next*.
4. From the *Specify Setup Mode* tab, select the default startup mode of this extension, specify the user preferences, and choose *Next*.
5. Choose *Finish* to close the *Company Assignment Wizard* window.
6. To check the assigned extensions in a company, in the *SAP Business One Extension Manager* window, choose the *Company Assignment* tab. From *Company List*, select the company and check whether the extension is available.

   For more information, see *Deploying Apps to a Company*.

6.3.3 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:

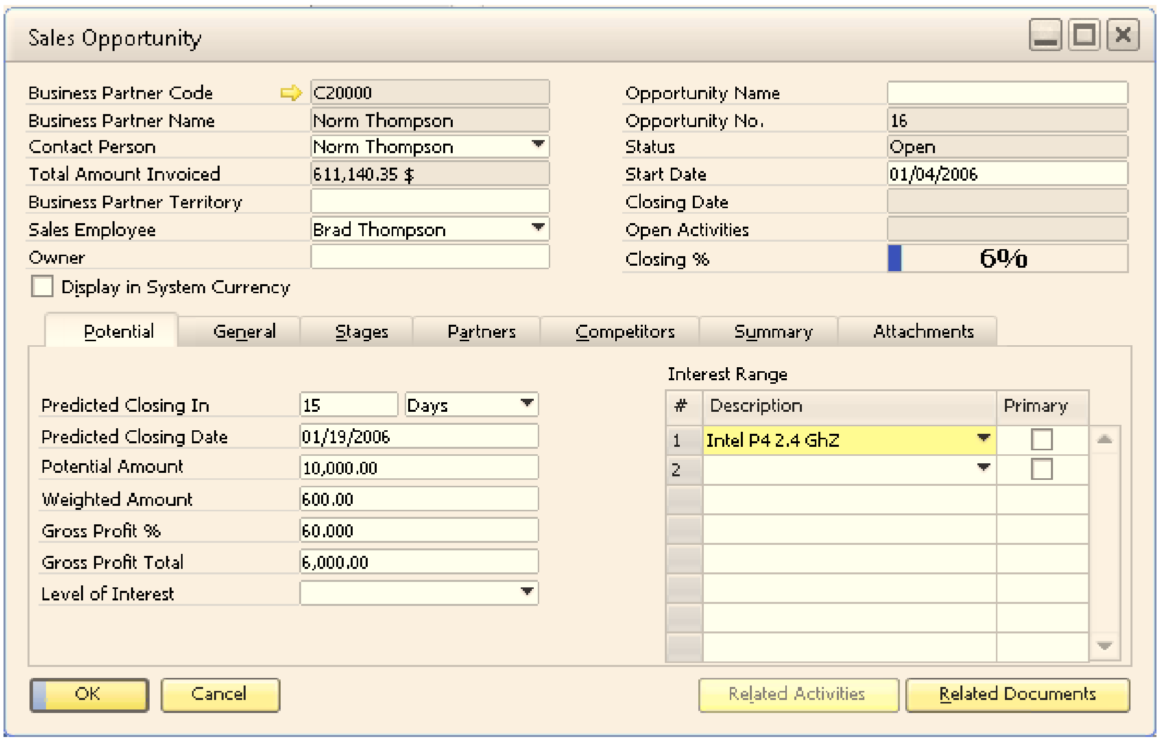
1. Under the OData folder, create another child folder and name it according to the company name.
2. Copy all the OData definition files (view and xsodata) to the folder.
3. Replace all placeholders `{COMPANY}` with the company name in all views and xsodata files.
4. Save, commit, and activate the OData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

6.3.4 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.

6.3.5 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.



Sales Opportunity

Business Partner Code	C20000	Opportunity Name	
Business Partner Name	Norm Thompson	Opportunity No.	16
Contact Person	Norm Thompson	Status	Open
Total Amount Invoiced	611,140.35 $	Start Date	01/04/2006
Business Partner Territory		Closing Date	
Sales Employee	Brad Thompson	Open Activities	
Owner		Closing %	6%

Display in System Currency

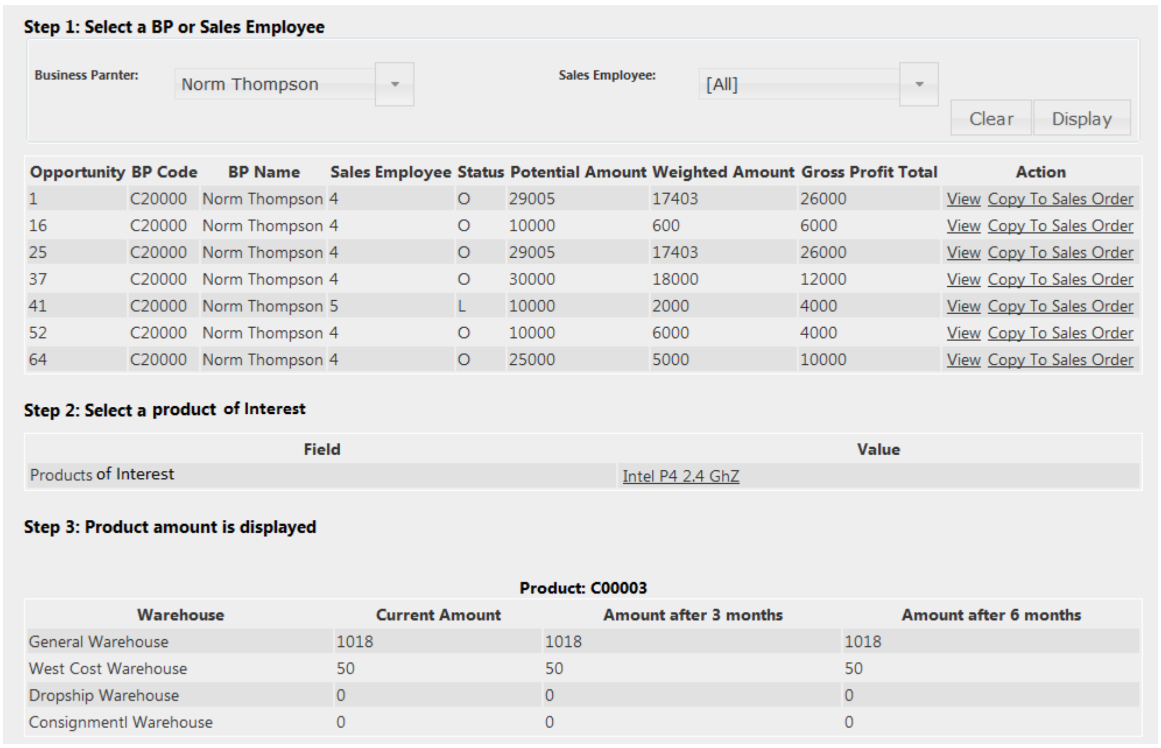
Predicted Closing In	15	Days
Predicted Closing Date	01/19/2006	
Potential Amount	10,000.00	
Weighted Amount	600.00	
Gross Profit %	60.000	
Gross Profit Total	6,000.00	
Level of Interest		

Interest Range

#	Description	Primary
1	Intel P4 2.4 GhZ	☐
2		☐

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

Business Parnter: Norm Thompson

Sales Employee: [All]

Opportunity	BP Code	BP Name	Sales Employee	Status	Potential Amount	Weighted Amount	Gross Profit Total	Action
1	C20000	Norm Thompson	4	O	29005	17403	26000	View Copy To Sales Order
16	C20000	Norm Thompson	4	O	10000	600	6000	View Copy To Sales Order
25	C20000	Norm Thompson	4	O	29005	17403	26000	View Copy To Sales Order
37	C20000	Norm Thompson	4	O	30000	18000	12000	View Copy To Sales Order
41	C20000	Norm Thompson	5	L	10000	2000	4000	View Copy To Sales Order
52	C20000	Norm Thompson	4	O	10000	6000	4000	View Copy To Sales Order
64	C20000	Norm Thompson	4	O	25000	5000	10000	View Copy To Sales Order

Step 2: Select a product of Interest

Field	Value
Products of Interest	Intel P4 2.4 GhZ

Step 3: Product amount is displayed

Product: C00003

Warehouse	Current Amount	Amount after 3 months	Amount after 6 months
General Warehouse	1018	1018	1018
West Cost Warehouse	50	50	50
Dropship Warehouse	0	0	0
Consignmentl Warehouse	0	0	0

www.sap.com/contactsap

Material Number