**Integration Framework**

**Version 2**

**APIs**

**Last Changed: Januar 4, 2020**

Table of Contents

[1 API Structure 4](#_Toc29036894)

[1.1 Introduction 4](#_Toc29036895)

[1.2 Calling Options for the API Services 5](#_Toc29036896)

[1.3 Generic Request Document 7](#_Toc29036897)

[1.4 Generic Response Document 7](#_Toc29036898)

[2 Deployment-Specific Properties APIs 8](#_Toc29036899)

[2.1 Creating Deployment-Specific Properties 9](#_Toc29036900)

[2.2 Deleting Deployment-Specific Properties 10](#_Toc29036901)

[2.3 Getting a List of Deployment-Specific Properties 11](#_Toc29036902)

[2.4 Getting Deployment-Specific Property Settings for All Deployments 12](#_Toc29036903)

[2.5 Setting Values for Deployment-Specific Properties 13](#_Toc29036904)

[3 Deployment APIs 14](#_Toc29036905)

[3.1 Creating a Deployment 14](#_Toc29036906)

[3.2 Deleting a Deployment 15](#_Toc29036907)

[3.3 Getting a List of Deployments 16](#_Toc29036908)

[3.4 Getting Deployment Details 17](#_Toc29036909)

[3.5 Getting Ports 18](#_Toc29036910)

[3.6 Setting Deployments 20](#_Toc29036911)

[3.7 Validating a Deployment 22](#_Toc29036912)

[3.8 Defining Overlay Properties 23](#_Toc29036913)

[3.9 Defining Inbound and Outbound Filter Definitions 24](#_Toc29036914)

[3.10 Getting Inbound and Outbound Filter Definitions 26](#_Toc29036915)

[3.11 Deleting Inbound and Outbound Filter Definitions 28](#_Toc29036916)

[3.12 Checking the Redeployment Status 28](#_Toc29036917)

[4 Customer APIs 29](#_Toc29036918)

[4.1 Adding Customers 29](#_Toc29036919)

[4.2 Deleting Customers 31](#_Toc29036920)

[4.3 Getting a Customer List 32](#_Toc29036921)

[4.4 Getting Customer Details 32](#_Toc29036922)

[4.5 Getting Customer Attachments 35](#_Toc29036923)

[4.6 Updating Customers 36](#_Toc29036924)

[5 Activation APIs 37](#_Toc29036925)

[5.1 Activating a Deployment 37](#_Toc29036926)

[5.2 Deactivating a Deployment 37](#_Toc29036927)

[5.3 Deactivating All Deployments 37](#_Toc29036928)

[5.4 Unsubscribing Scenario Steps 38](#_Toc29036929)

[5.5 Batch Mode for Scenario Package Activation 39](#_Toc29036930)

[5.6 Batch Mode for Scenario Package Deactivation 40](#_Toc29036931)

[5.7 Getting Status Information for Scenario Package Deployments 41](#_Toc29036932)

[6 System Landscape Directory (SLD) APIs 42](#_Toc29036933)

[6.1 Getting SLD Entries 42](#_Toc29036934)

[6.2 Testing Connections 44](#_Toc29036935)

[6.3 Adding SLD Entries 45](#_Toc29036936)

[6.4 Updating SLD Entries 47](#_Toc29036937)

[6.5 Deleting SLD Entries 49](#_Toc29036938)

[6.6 Copying SLD Entries 49](#_Toc29036939)

[6.7 Getting SLD Entry Details 50](#_Toc29036940)

[6.8 Getting System Types and Connectivity Types 52](#_Toc29036941)

[6.9 Getting System Type Properties 52](#_Toc29036942)

[6.10 Getting SLD Categories 53](#_Toc29036943)

[6.11 Creating an SLD Category 54](#_Toc29036944)

[6.12 Deleting an SLD Category 54](#_Toc29036945)

[6.13 Setting Categories for SLD Entries 55](#_Toc29036946)

[7 Transaction APIs 56](#_Toc29036947)

[7.1 Getting Transaction Information 56](#_Toc29036948)

[8 Scenario Package Configuration APIs 60](#_Toc29036949)

[8.1 Getting the Scenario Package Configuration 60](#_Toc29036950)

[8.2 Deleting a Scenario Package 60](#_Toc29036951)

[9 Internal Queue APIs 62](#_Toc29036952)

[9.1 Getting the Number of Messages in a Queue 62](#_Toc29036953)

[9.2 Getting Information About Messages in the Queue 63](#_Toc29036954)

[9.3 Getting Messages with IDs and Trace IDs from Queue 66](#_Toc29036955)

[9.4 Getting a Message from a Queue 67](#_Toc29036956)

[9.5 Deleting a Message from a Queue 68](#_Toc29036957)

[9.6 Clearing Queues 69](#_Toc29036958)

[10 User Administration APIs 69](#_Toc29036959)

[10.1 Getting a User List 69](#_Toc29036960)

[10.2 Creating Users 70](#_Toc29036961)

[10.3 Deleting Users 72](#_Toc29036962)

[11 Cockpit APIs 73](#_Toc29036963)

[11.1 Getting Cockpit Results 73](#_Toc29036964)

[12 Transaction Protocol APIs 75](#_Toc29036965)

[12.1 Getting Aggregated Values for a Day, Month or Year 75](#_Toc29036966)

[12.2 Getting Series of Aggregated Values for a Period 77](#_Toc29036967)

[Appendix 79](#_Toc29036968)

[1 Status Messages 79](#_Toc29036969)

[1.1 General Messages 79](#_Toc29036970)

[1.2 Messages of Deployment-Specific Properties Service 80](#_Toc29036971)

[1.3 Deployment Service Messages 80](#_Toc29036972)

[1.4 Customer Service Messages 81](#_Toc29036973)

[1.5 Activation Service Messages 82](#_Toc29036974)

[1.6 SLD Service Messages 82](#_Toc29036975)

[1.7 Transaction Service Messages 83](#_Toc29036976)

[1.8 Scenario Packages Service Messages 83](#_Toc29036977)

[1.9 Internal Queue Service Messages 84](#_Toc29036978)

[1.10 Cockpit Service Messages 84](#_Toc29036979)

[1.11 Transaction Protocol Service Messages 84](#_Toc29036980)

[1.12 User Service Messages 84](#_Toc29036981)

[2 SLD System Type Properties and Default Values 85](#_Toc29036982)

[2.1 Database System (J.AnySystem) 85](#_Toc29036983)

[2.2 SAP Business One Systems (B1.2004, B1.2005, B1.2007, B1.8.8, B1.9.0) 85](#_Toc29036984)

[2.3 SAP Business Warehouse Systems (BI Systems (BI.3.5.3, BI.7.0.3)) 86](#_Toc29036985)

[2.4 E-Mail System (E.AnySystem) 86](#_Toc29036986)

[2.5 HTTP System (H.AnySystem) 87](#_Toc29036987)

[2.6 File Systems (F.AnySystem, F.B1.2004, F.B1.2005, F.B1.2007) 87](#_Toc29036988)

[2.7 SAP ERP Systems, WS Sections Only for ECC 6.0 87](#_Toc29036989)

[2.8 FTP System (P.AnySystem) 89](#_Toc29036990)

[2.9 Timer System (T.AnySystem) 90](#_Toc29036991)

[Copyrights, Trademarks, and Disclaimers 91](#_Toc29036992)

# 1 API Structure

## 1.1 Introduction

The integration framework API offers services to work with deployment-specific properties, deployments, customer entries, deployment activation and deactivation, System Landscape Directory entries, integration framework transactions, package configuration, internal queues, users cockpit and transaction protocol information.

To use the API, do the following:

|  |  |
| --- | --- |
| Step | Description |
| 1. Use the general URL to access the API or use the general BizFlow for internal consumption. | See the next section |
| 1. Define the URL parameters | Hand over the following information:   * Payload type definition. The API offers the option to either use XML or JSON. * Scenario package identifier (if required by the service) * Deployment identifier (if required by the service)   See the next section |
| 1. Use the general request structure. | In the request structure, define the method and the mode.  See section 1.3 Generic Request Document |
| 1. For the method and mode, refer to the related section. | In the related section, look up the request details. |

**Example for Adding a Deployment-Specific Property Calling the API from Outside**

**Access URL**

http://127.0.0.1:8080/B1iXcellerator/exec/ipo/iFw2.APIs/com.sap.b1i.dev.api/ipo/api2.ipo/api.caller?bpm.pltype=xml&ipac=sap.test

**Access User and Authentication**

For an incoming HTTP call, you must enter a user name and password for authentication. Use an active runtime user name and password. Note that the runtime user needs access rights to the API. To enable access, choose Maintenance → User Administration, click Runtime Users and select the second checkbox in the Status column.

**XML Request**

<api.call method="props" mode="add">

<request>

<entry>

<id>test1</id>

<desc>apitest</desc>

<default>123</default>

<type>String</type>

</entry>

</request>

</api.call>

**XML Response**

<api.response version="1.0" ts="2018/09/19 16:42:32">

<status id="000" desc="1 entries added"/>

<par method="props" mode="ass" ipac="sap.test" deplid="" pltype="xml" incl="props.bfd"/>

<response>

<entry id="test1” status="1.000" desc="OK"/>

</response>

</api.response>

## 1.2 Calling Options for the API Services

You can call the API services by an incoming HTTP call into the integration framework or by including a BizFlow into a scenario step.

**Using APIs Through Incoming Calls**

When calling the APIs from outside, use the following general service to call all APIs:

On premise:

http(s)://<ipaddress>:<port>/B1iXcellerator/exec/ipo/iFw2.APIs/com.sap.b1i.dev.api/ipo/api2.ipo/api.caller?<urlpar>

Cloud:

https://<logicalCloudAddress>/<PaasTenantIdentifier>/B1iXcellerator/exec/ipo/iFw2.APIs/com.sap.b1i.dev.api/ipo/api2.ipo/api.caller?<urlpar>

|  |  |  |
| --- | --- | --- |
| Parameter |  | Description |
| ipaddress |  | IP address of integration framework server |
| port |  | Integration framework port, default port for http is 8080, default port for https is 8443 |
| logicalCloudAddress |  | Logical cloud address |
| PaasTenantIdentifier |  | Tenant identifier, if deployed in the cloud |
| urlpar | ipac | Scenario package identifier, parameter is optional for services that do not require the identifier. |
| deplid | Optional parameter for deployment identifier. Depending on the method or mode, the parameter can be mandatory |
| bpm.pltype | Set the payload type. You can use the API service in xml format or in json format. The parameter is mandatory.  The payload type setting for the request determines the payload type of the response. |

When calling the APIs from outside, the call is secured through basic authentication. Provide credentials of a runtime user. The runtime user must be active and must have access rights to the API.

**Internal Consumption of API Services**

For internal consumption, directly call the API BizFlow and set the payload type to XML (bpm.pltype=xml):

/com.sap.b1i.dev.api/bfd/trigger.bfd?!<urlpar>

<urlpar> parameters have the following names:

* querykey.ipac
* querykey.deplid
* querykey.bpm.pltype

**Example**

Request all SLD entries with B1.9.0 system type:

* Create a scenario step, for example, with timer inbound and void outbound.
* For the processing, add an XSL transformation atom that defines the XML request document.
* Add an include atom after the transformation atom.
* Add the sequence flows and save the step.

**XSL Transformation**

In the XSL file, add the following to <xsl:template match="/">:

<api.call method="sld" mode="getlist">

<request>

<systype>B1.9.0</systype>

</request>

</api.call>

**Include**

In the include atom, enter the following information:

* Type: Bizflow
* Link to Sources: /com.sap.b1i.dev.api/bfd/trigger.bfd (Fixed Value)

Optionally add the payload type parameter as a property to the Properties section of the include atom in the following way:

Identifier: querykey.bpm.pltype Value: xml

Open the test environment, activate the step and click Run.

Click Debug to display the result.

In the flow debugger, click the red arrow of the EndBizFlow element. The framework displays the following response:

<api.response version="1.0" ts="2019/05/15 13:31:41">

<status id="000" desc="OK"/>

<par method="sld" mode="getlist" ipac="" deplid="" pltype="xml" incl="sld.bfd"/>

<response>

<list filter="systype='B1.9.0'">

<entry sysid="0010000100" systype="B1.9.0" name="SBODEMODE"/>

<entry sysid="0010000101" systype="B1.9.0" name="SBODEMOSMB"/>

<entry sysid="0010000104" systype="B1.9.0" name="SAPCALIOT"/>

<entry sysid="0010000110" systype="B1.9.0" name="SBODEMOAR"/>

<entry sysid="0010000115" systype="B1.9.0" name="SBOSUP"/>

<entry sysid="001sap0005" systype="B1.9.0" name="SBO-COMMON"/>

</list>

</response>

</api.response>

**Note**: If the API method requires the ipac and deplid parameters, add the parameters as properties to the Properties section of the include atom in the following way:

Identifier: querykey.ipac Value: <name of package>

Identifier: querykey.deplid Value: <name of deployment>

## 1.3 Generic Request Document

To request information, use the following generic request document structure.

**XML Format**

<api.call method="val" mode="val">

<request>

...

</request>

</api.call>

**JSON Format**

{

"method": "val",

"mode": "val",

"request": { … }

}

|  |  |
| --- | --- |
| Parameter | Description |
| method | Mandatory method identifier parameter, for example, props |
| mode | Mandatory mode parameter for the method, for example, add |
| request | Optional request structure of method and mode.  For more information, see the sections below |

## 1.4 Generic Response Document

The API responses using the following document structure:

**XML Response**

<api.response version="val" ts="val">

<status id="val" desc="val"/>

<par method="val" mode="val" ipac="val" deplid="val" pltype="val" incl="val"/>

<response>

...

</response>

</api.response>

**JSON Response**

{

"version": "val",

"ts": "val",

"status": { "id": "val", "desc": "val" },

"par": { "method": "val", "mode": "val", "ipac": "val", "deplid": "val", "pltype": "val", "incl": "val" },

"response": { … }

}

| Parameter | Description |
| --- | --- |
| version | API version, for example, 1.0 |
| ts | Formatted timestamp of response, for example, 2018/08/23 15:27:27 |
| status/id | Status of call, 000 for a successful call, otherwise with an error number.  For more information about error numbers, see the appendix |
| status/desc | Long text of call status |
| par/method | Echo of method inbound parameter |
| par/mode | Echo of mode inbound parameter |
| par/ipac | Echo of ipac URL parameter |
| par/deplid | Echo of deplid URL parameter |
| par/pltype | Echo of pltype URL parameter |
| par/incl | Name of internal BizFlow that provides the called API service |
| response | Optional response structure of method and mode.  For more information, see the sections below |

# 2 Deployment-Specific Properties APIs

The API service allows you to add or delete deployment-specific properties in scenario package design. You can set values for available deployment-specific properties when setting up the deployment. You can retrieve a list of defined deployment-specific properties or the defined values for a deployment.

| Mode Options | Description |
| --- | --- |
| mode=add | Add one or many deployment-specific properties |
| mode=del | Delete one or more deployment-specific properties |
| mode=get | Get a list of defined deployment-specific properties |
| mode=getval | Get a list of deployment-specific property values for one or all deployments |
| mode=set | Set the value for one or many deployment properties in the setup for a deployment |

## 2.1 Creating Deployment-Specific Properties

Create one or multiple deployment-specific properties in scenario package design. The properties are global properties that are valid for the package. The API adds correct and skips incorrect entries. The response document contains all details. The service also adds the properties to the setup in deployment.

method=props

mode=add

**XML Request**

<request>

<entry>

<id>val</id>

<desc>val</desc>

<enum>val</enum>

<pwd>val</pwd>

<default>val</default>

<type>val</type>

</entry>

...

</request>

**JSON Request**

"request": {

"arr": [

{"id":"val","desc":"val","enum":"val","pwd":"val","default":"val","type":"val"},

…

]

}

| Parameter | Description |
| --- | --- |
| id | Mandatory name or identifier of the property. Id must follow the standards of the naming convention |
| desc | Optional description of the property displayed as a tooltip in the deployment phase |
| enum | Optional list of valid values. In the deployment phase, the enumerations are displayed in a drop-down list. Define values separated by \*, for example, 1\*2\* |
| pwd | Optional Boolean parameter (true/false) to indicate whether the property is a password. In the deployment phase, such a password property is displayed with \*and stored encrypted. The default is false |
| default | Optionally define a default value that is used if the value is not set in the deployment phase |
| type | Optionally define the data type of the property. The default is String. The integration framework also supports Array of Strings. |

**XML Response**

<response>

<entry id="val" status="val" desc="val"/>

...

</response>

**JSON Response**

"response": {

"arr": [

{ "id":"val", "status":"val", "desc":"val" },

…

]

}

}

| Parameter | Description |
| --- | --- |
| id | Name or identifier of the property |
| status | Status. 1.000 for success, otherwise an error number.  For more information, see the appendix |
| desc | Long text for status |

## 2.2 Deleting Deployment-Specific Properties

Delete one or multiple deployment-specific properties in scenario package design. If properties and values are already defined in the deployment phase, the API also deletes the properties from all deployments. The API deletes all array elements for properties of Array of Strings type. The API deletes correct entries and skips incorrect entries. For more information, see the response document

method=props

mode=del

**XML Request**

<request>

<entry>

<id>val</id>

</entry>

...

</request>

**JSON Request**

"request": {

"arr": [ { "id": " val" }, … ]

}

| Parameter | Description |
| --- | --- |
| id | Name or identifier of the mandatory property |

**XML Response**

<response>

<entry id="val" status="val" desc="val"/>

...

</response>

**JSON Response**

{

"response": {

"arr": [

{ "id":"val", "status":"val", "desc":"val" },

…

]

}

}

| Parameter | Description |
| --- | --- |
| id | Name or identifier of the property |
| status | Individual status of entry. 1.000 for success, otherwise error number.  For more information, see the appendix |
| desc | Long text of entry status |

## 2.3 Getting a List of Deployment-Specific Properties

Get a list of deployment-specific properties for a scenario package.

method=props

mode=get

**Request**

A request document is not required.

**XML Response**

<response>

<entry id="val" desc="val" enum="val" pwd="val" default="val" type="val" b1iftype="val" gen="val"/>

...

</response>

**JSON Response**

"response": {

"arr": [

{"id":"val","desc":"val","enum":"val","pwd":"val","default":"val","type":"val"},

…

]

}

## 2.4 Getting Deployment-Specific Property Settings for All Deployments

Get a list of deployment-specific properties with values for all deployments of a scenario package.

method=props

mode=getval

**Request**

A request document is not required.

If you leave the deplid URL parameter empty, the API returns property values for all deployments, otherwise the API returns property values for the deployment.

**XML Response**

<response>

<depl id="val">

<prop id="val">

<index id="1">val</entry>

…

</prop>

<prop id="val">\*\*\*\*\*\*\*\*\*\*</prop>

<prop id="val">val</prop>

…

</depl>

…

</response>

| Parameter | Description |
| --- | --- |
| depl | Values for a concrete deployment, attribute id is deployment identifier |
| prop | Property with value. Attribute id is property identifier. The value is in element body. If property is a password, the API returns the value \*\*\*\*\*\*\*\*\*\*. If the property is of Array of Strings type, the API lists the values with the index element |

**JSON Response**

"response": {

"arr": [

{ "deplid":"val",

"props": [

{ "prop": "val", "arrval": [ {"index.1": "val"}, … ] },

{ "prop": "val", "value": "\*\*\*\*\*\*\*\*\*\*" },

{ "prop": "val", "value": "val" },

…

]

}

…

]

}

| Parameter | Description |
| --- | --- |
| deplid | Deployment identifier |
| props | Array of objects with property values for the deployment with the deplid identifier |
| prop | Property name |
| value | Property value. If the property is a password, the API returns the \*\*\*\*\*\*\*\*\*\* value |
| arrval | If property is Array of Strings type, the API lists the values in an array. If a property is a password, the API returns the \*\*\*\*\*\*\*\*\*\* value |

## 2.5 Setting Values for Deployment-Specific Properties

Set values for one or multiple deployment properties for one or multiple deployments.

method=props

mode=set

**Request**

If the deplid URL parameter is empty, the API sets property values for all deployments, otherwise the API sets the value for the specified deployment. If the property is a password, the API encrypts the value. If a property is of Array of Strings type, the API adds a new value to the property.

**XML Request**

<request>

<entry id="id">val</entry>

<entry id="id" op="del">val</entry>

…

</request>

**JSON Request**

{

"request": {

"arr": [

{ "id":"val", "value":"val", ("op":"val") },

…

]

}

}

| Parameter | Description |
| --- | --- |
| id | Property identifier |
| val | Property value |
| op | Optional, if the property is of Array of Strings type, op (add, del) determines, if the API adds or deletes the array element with the value. op=add is the default. |

**XML Response**

<response>

<entry id="val" status="val" desc="val" op="val"/>

…

</response>

**JSON Response**

{

"response": {

"arr": [

{"id":"val", "status":"val", "desc":"val ", "op": "val" }, …

]

}

| Parameter | Description |
| --- | --- |
| id | Property identifier |
| status | Status. 1.000 for success, otherwise the error number.  For more information, see the appendix. |
| desc | Status long text |
| op | Operation the API performed.  - if an error occurs  set if the API set the value  add if the API added the value of property of Array of Strings type  del if the API removed all values of property of Array of Strings type |

# 3 Deployment APIs

The API allows you to work with deployments for the integration framework version 2. You can retrieve deployment information, add or delete a deployment of a scenario package and set deployment definitions.

| Deployment Options | Description |
| --- | --- |
| mode=add | Add a deployment |
| mode=del | Delete one or more deployments |
| mode=getlist | Get a list of deployments |
| mode=get | Get information about a deployment |
| mode=getport | Get a list of ports that require SLD assignments |
| mode=set | Set a deployment definition value |
| mode=validate |  |

## 3.1 Creating a Deployment

Create a deployment for a scenario package.

method=depl

mode=add

The deplid URL parameter is mandatory and you can only use alpha-numeric characters. If the deployment already exists, the API does not add the deployment. If the integration framework runs in single tenancy mode, you can only add one deployment. The API does not consider the deplid URL parameter and the deployment identifier is set to ST (single tenant). The integration framework initializes defined deployment-specific properties for the deployment phase.

**Request**

The request document is optional.

**XML Request**

<request>

<cust.id>val</cust.id>

</request>

**JSON Request**

"request": {

"cust.id":"val"

}

| Parameter | Description |
| --- | --- |
| cust.id | Optional technical identifier (GUID) of the associated customer. You can retrieve a list of technical customer identifier using the customer API, see below |

**XML Response**

<response>

<deplid>val</deplid>

<multitenancy>val</multitenancy>

<id>val</id>

</response>

**JSON Response**

"response": { "deplid":"val", "multitenancy":"val", "id":"val" }

| Parameter | Description |
| --- | --- |
| deplid | Deployment identifier |
| multitenancy | Set to true, the integration framework runs in multi-tenancy mode |
| id | Generated technical deployment identifier (32-byte GUID) |

## 3.2 Deleting a Deployment

Delete a deployment for a scenario package. If scenario steps are active, the integration framework rejects the deletion request. Use the activation APIs to deactivate the scenario steps for the deployment.

The API deletes the following artefacts:

* Deployment
* Deployment definition
* Settings for deployment-specific properties
* Deployment-related user interface settings
* Optional subscriptions to SAP Business One
* Optional subscriptions to SAP ERP
* Generated BizFlows

method=depl

mode=del

**Request**

A request document is not required. The deplid URL parameter is mandatory.

**XML Response**

<response>

<depl.id>val</depl.id>

<actions> x adjustments in deployment and x bizflows removed</actions>

</response>

**JSON Response**

"response": {

"id": "val",

"actions": "x adjustments in deployment and x bizflows removed"

}

| Parameter | Description |
| --- | --- |
| id | Technical deployment identifier of removed deployment (32-byte GUID) |
| actions | Performed actions |

## 3.3 Getting a List of Deployments

Get a list containing all deployments of a scenario package.

method=depl

mode=getlist

**Request**

A request document is not required.

**XML Response**

<response>

<depl name="val" id="val" setup="val" valid="val" active="val"/>

…

</response>

**JSON Response**

"response": {"arr": [

{ "name":"val", "id":"val", "setup":"val", "valid":"val", "active":"val" },

…

]

}

| Parameter | Description |
| --- | --- |
| name | Deployment identifier |
| id | Technical deployment identifier |
| setup | Indicates whether the deployment is set up. Possible values are true and false |
| valid | Indicates whether the deployment setup is valid.  If setup=’false’, the value is –  Otherwise, the value is true or false |
| active | Indicates whether the deployment is active |

## 3.4 Getting Deployment Details

Get details for a deployment of a scenario package.

method=depl

mode=get

**Request**

A request document is not required.

**XML Response**

<response>

<id>val</id>

<setup>val</setup>

<valid>val</valid>

<active>val</active>

<step.setup.list>

<step scen="val" id="val">

<port name="val" type="val" connType="val">

<sysid>val</sysid>

…

</port>

…

</step>

…

</step.setup.list>

</response>

**JSON Response**

"response": {

"id": "val",

"setup": "val",

"valid": "val",

"active": "val",

"step.setup.list": [

{

"scen": "val",

"id": "val",

"port.list": [

{

"name": "val",

"type": "val",

"connType": "val",

"sysid.list": [ { "sysid":"val", … } ]

},

…

]

},

…

]

}

| Parameter | Description |
| --- | --- |
| id | Technical deployment identifier |
| setup | Indicates whether the deployment is set up. Possible values are true and false |
| valid | Indicates whether the deployment setup is valid.  If setup=’false’, the value is –  Otherwise, the value is true or false |
| active | Indicates whether the deployment is active |
| step.setup.list | List of steps that are set up |
| scen | Scenario name |
| id | Scenario step name |
| port.list | List of ports that are set up |
| name | Port name |
| type | Port type, either S for single port, or G for group port (local or global in the user interface) |
| connType | Connectivity type of port, for example, B1EV for SAP Business One by Event |
| sysid.list | List of associated SLD SysId entries |

## 3.5 Getting Ports

The API returns a list of scenarios and steps of a scenario package, and for each step, the API lists all ports that require an SLD assignment. You need the information for setting up the deployment using the set API. For more information, see the next section.

**Prerequisites**

The scenario package and the deployment must exist. You can add a deployment using the add mode of the service.

method=depl

mode=getport

**Request**

A request document is not required.

**XML Response**

<response>

<scenario id=" val">

<step id="val">

<port ctype="val" ptype="val" name="val" connType="val"/>

…

</step>

…

</scenario>

…

</response>

**JSON Response**

"response": {

"scenarios": [

{ "id": "sap.scen01",

"steps": [

{ "id": "sap.step01",

"ports": [

{ "ctype": "Ip",

"ptype": "G",

"name": "B1InboundG",

"connType": "B1EV"

},

…

]

},

…

},

…

]

}

| Parameter | Description |
| --- | --- |
| scenario/id | Scenario identifier |
| step/id | Scenario step identifier |
| ctype | Call type.  Ip for primary inbound  IOp for primary in/outbound  Op for primary outbound  IOs for secondary in/outbound. |
| ptype | Port type.  S is single port (local in the user interface)  G is group port (global in the user interface) |
| name | Port identifier |
| connType | Connectivity type. Adapter name, for example, B1EV SAP Business One by Event |

## 3.6 Setting Deployments

Define scenario steps for activation and for associating SLD SysId entries to ports. To retrieve all scenarios and scenario steps of the package with all mandatory port assignments, use the getport mode of the service. See the previous section

**Prerequisites**

* The scenario package and deployment must exist. If a deployment does not exist, use the add mode of the service. See section 3.1
* The API adds the deployment without activation. For more information about activation, see section 5.1
* For the activation, the deployment must be valid. To check the validation, use the validate mode of the service. For more information, see section 3.7
* The service offers several validation checks. If an error occurs, the integration framework rejects the API task. The API returns the first error.

method=depl

mode=set

**Request**

The request document is optional.

**XML Request**

<request mode="val" debug="val">

<scenario name="val" debug="val">

<step name="val"/>

…

</scenario>

…

<port name="val">

<entry>val</entry>

…

</port>

…

<request>

**JSON Request**

"request": {

"mode":"override",

"debug":"true",

"scenarios": [

{ "id":"sap.scen01",

"debug":"true",

"steps": [ {"id":"val"}, … ]

}

],

"ports": [

{ "id":"B1CallG", "entries": [ {"id":"val"}, … ] },

…

]

}

| Parameter | Description |
| --- | --- |
| mode | If mode=’create’, the integration framework rejects the deployment, if the deployment is already set up.  Use override to create the deployment and replace the existing one |
| debug | Optional general debug setting. Possible values are true or false, the default is false. The API uses the default value, if scenarios are not explicitly defined or if the debug definition is not provided. |
| scenario | Optional definition.  If only selected scenarios are going to be part for a later activation, define the scenarios. If <scenario> is not defined, the API marks all scenarios and all scenario steps for a later activation |
| scenario/name | Scenario identifier |
| scenario/debug | Optional debug setting to overlay the general debug setting. Possible values are true or false. The default is the general setting |
| step | Optional definition. If only selected scenario steps are part of a later activation, define the scenario steps. If <step> is not defined in the scenario, the API marks all scenario steps of the scenario for a later activation.  The API adds error steps for later activation automatically, if you do not define the explicitly |
| step/name | Scenario step identifier |
| port | Definition of port assignments |
| port/name | Port identifier, either a group or single identifier |
| entry | SLD SysId entry to assign to the port.  The API validates SysId entries and checks whether the entries are correct, whether they are available in SLD and whether the type fits the assigned port.  For multi ports (B1EV, CRON, FILI, MQIN, B1AO.1), you can assign more than one SysId. |

**XML Response**

<response>

<deplid>val</deplid>

<id>val</id>

<mode>val</mode>

<multitenancy>val</multitenancy>

</response>

**JSON Response**

"response": {

"deplid": "val",

"id": "val"

"mode": "val"

"multitenancy": "val",

}

| Parameter | Description |
| --- | --- |
| deplid | Deployment identifier |
| id | Internal technical deployment identifier (32-byte GUID) |
| mode | Either override or create |
| multitenancy | If true, the integration framework runs in multi-tenancy mode |

## 3.7 Validating a Deployment

Validate a deployment of a scenario package. A valid deployment is mandatory for activation. If the API returns a negative validation result, the API service provides a detailed error report.

method=depl

mode=validate

**Request**

The request document is not required.

If the deplid URL parameter is empty, the validation runs checking all deployments.

**XML Response**

<response>

<validation id="val" status="val">

<err scen="val" step="val" type="val" port="val" adapter="val" prop="val">val</err>

…

</validation>

…

</response>

**JSON Response**

"response": {

"deployments": [

{ "id": "val",

"status": "val",

"error.report": [

{"scen":"val", "step":"val", "type":"val", "portid":"val", "adapter":"val", "prop":"val",  
 "desc": "val"},

…

]

}

…

]

}

| Parameter | Description |
| --- | --- |
| id | Deployment identifier |
| status | Validation status, either true or false |

**Optional Error Report**

| Parameter | Description |
| --- | --- |
| scen | Scenario identifier where the error is located |
| step | Scenario step identifier of scenario step where the error is located |
| type | Connectivity type.  Ip for primary inbound  IOp for primary in/outbound  Op for primary outbound  IOs for secondary inbound/outbound |
| portid | Identifier of single or group port in which the error occurs |
| adapter | Adapter type for which the error occurs, for example, B1EV |
| prop | Property identifier where the error occurs |
| desc | Error description |

## 3.8 Defining Overlay Properties

Define overlays for property definitions in scenario design. The service creates the sysid.setup.xml document in the setup scenario folder of the BizStore. The service merges definitions into an already existing document. The ipac package identifier and the deplid deployment identifier URL parameters are mandatory.

method=depl

mode=setoverlay

**XML Request**

<request>

<scenario id="val">

<step id="val">

<sysid id="val">

<overlay prop="val" link="val">val</overlay>

...

</sysid>

...

</step>

...

</scenario>

...

</request>

**JSON Request**

"request": {

"scen":[

{ "id":"val",

"step":[

{ "id":"val",

"sysid":[

{ "id":"val",

"overlay":[ {"id":"val", "link":"val", "val":"val"},… ]

} …

]

} …

] } …

] }

}

| Parameter | Description |
| --- | --- |
| scen | Array of scenario definitions (id=identifier) |
| step | Array of Scenario step definitions (id=identifier) |
| sysid | Array of SLD system identifiers (id=identifier) |
| overlay | Array of overlay definitions (id=name of property, val=property value) |
| link | Optional link to a label that is defined in package configuration in the Inbound Property Overlay function.  If it is not possible to resolve the link, the service takes the body values |

**Response**

There is no response document.

## 3.9 Defining Inbound and Outbound Filter Definitions

Define filters for inbound and outbound processing for scenario packages developed with framework version 2. The following adapters currently support filter definitions:

* Inbound B1EV
* Inbound IDOC
* Outbound B1AO.1

You use rules to define filters for a scenario step. Rules apply to a sender or sender and receiver system combination. You can define multiple rules for a scenario step.

A rule consists of one or more lines combined by the logical AND or OR operator.

The API replaces all already available rules (all lines) by lines defined in the API request document.

method=depl

mode=setfilter

**XML Request**

<request>

<scenario id="val">

<step id="val">

<rule type="val" sender="val" receiver="val">

<line pos="val" andor="val" crit="val" val1="val" val2="val"   
 active="val">val</line>

...

</rule>

...

</step>

...

</scenario>

...

</request>

**JSON Request**

"request": {

"scen":[

{ "id":"val",

"step":[

{ "id":"val",

"rule":[

{"type":"val","sender":"val","receiver":"val",

"line":[

{"pos":"val","andor":"val","crit":"val","op":"val",  
 "val1":"val","val2":"val","active":"val"},…

]

} …

]

} …

] } …

] }

}

| Parameter | Description |
| --- | --- |
| scen | Array of scenario definitions (id=identifier) |
| step | Array of scenario step definitions (id=identifier) |
| rule | Array of rules |
| type | Rule type. Use sender for rules running during the inbound phase or receiver for rules processed during the outbound phase. |
| sender | SLD SysId of the sender system |
| receiver | SLD SysId for the receiver system. The parameter is mandatory for the receiver type |
| line | Array of line definitions. A rule consists of one or multiple lines. The lines are ordered by pos and combined with the andor logical operator |
| pos | Number that defines the position of the line in the rule, starting with 1 |
| andor | Logical operator to add a line to the rule. The parameter is mandatory except for the first line. The operator options are AND or OR |
| crit | Name of criteria field. The criteria field definition must be part of the scenario package |
| op | Operator for the rule. Possible values are equal to, not equal to, greater than, less than, between, not between, greater than or equal to, less than or equal to, is in, contains, equal to blank, is not blank |
| val1 | Rule value for all operators that require a value, such as equal to and the first value for all rules that require two values, such as between. Not relevant for operators that do not require a value, such as equal to blank |
| val2 | Second rule value for operators that require two values, such as between. Not relevant for operators that need no or one value. |
| active | Checkbox to activate or deactivate a rule line. Either true or false |

**Response**

There is no response document.

## 3.10 Getting Inbound and Outbound Filter Definitions

Get filter definitions for a deployment of a scenario package.

method=depl

mode=getfilter

**XML/JSON Request**

There is no request document.

**XML Response**

<request>

<scenario id="val">

<step id="val">

<rule type="val" sender="val" receiver="val">

<line pos="val" andor="val" crit="val" val1="val" val2="val"   
 active="val">val</line>

...

</rule>

...

</step>

...

</scenario>

...

</request>

**JSON Response**

"response": {

"scen": [{

"id": "sap.Inbound.Scenarios",

"step": [{

"id": "sap.B1.Inbound",

"rule": [{

"type": "sender",

"sender": "0010000101",

"receiver": "",

"line": [{

"pos": "1",

"andor": "",

"crit": "Mandant",

"op": "equal to",

"dftStep": "Inbound/Outbound Message",

"val1": "102",

"val2": "",

"active": "false"

}, ...]

}, ...]

}, ...]

}, ...]

}

| Parameter | Description |
| --- | --- |
| scen | Array of scenario definitions (id=identifier) |
| step | Array of Scenario step definitions (id=identifier) |
| rule | Array of rules |
| type | sender rule type runs during the inbound phase and receiver rule type runs during outbound phase |
| sender | SLD SysId of the sender system |
| receiver | SLD SysId of the receiver system. Empty for sender type |
| line | Array of line definitions. A rule consists of one or multiple lines. The lines are ordered by pos and combined with andor logical operator |
| pos | Number defining the position of the line in the rule, beginning with 1 |
| andor | Logical operator to add a line to the rule. Empty for postion 1. Values are AND or OR |
| crit | Criteria field name. Criteria fields are defined for the scenario package |
| op | Rule operator. Possible values are equal to, not equal to, greater than, less than, between, not between, greater than or equal to, less than or equal to, is in, contains, equal to blank, is not blank |
| val1 | Rule value for all operators that require a value, such as equal to and the first value for all rules that require two values, such as between. Not relevant for operators that do not require a value, such as equal to blank |
| val2 | Second rule value for operators that require two values, such as between. Not relevant for operators that need no or one value. |
| active | Checkbox to activate or deactivate a rule line. Either true or false |

## 3.11 Deleting Inbound and Outbound Filter Definitions

Delete all filter definitions for a deployment of a scenario package. You can set a scenario, scenario step of a rule to only delete parts of the filter definitions for a deployment.

method=depl

mode=delfilter

**XML Request**

<request>

<scenario>val</scenario>

<step>val</step>

<rule type="val" sender="val" receiver="val"/>

</request>

**JSON Request**

"request":{"scen":"val","step":"val","rule":{"type":"val","sender":"val","receiver":"val"}}

| Parameter | Description |
| --- | --- |
| scen | Optional parameter to delete filters for a scenario. If empty, the API deletes the filters of the deployment |
| step | Optional parameter to delete filters for a scenario step. To use the parameter, setting the scen parameter is mandatory. |
| rule | Optional parameter to delete a rule for a scenario step. To use the parameter, setting scen and step parameters is mandatory. |
| type | Rule type. Use sender for rules running during the inbound phase or receiver for rules processed during the outbound phase. |
| sender | SLD SysId of the sender system |
| receiver | SLD SysId for the receiver system. The parameter is mandatory for the receiver type |

**Response**

There is no response document.

## 3.12 Checking the Redeployment Status

Check the status of a redeployment. The framework triggers redeployment with the activation or deactivation of scenario packages. During redeployment connectivity is locked. When triggering activation or deactivation through the API, API method allows you to check the status and ensures that you only trigger subsequent processes that require connectivity, if there is no redeployment running.

You can only call the method internally through a BizFlow definition (BFD), because HTTP connectivity is not available.

method=depl

mode=getredeplstatus

**Request**

There is no request document.

**XML Response**

<response>

<inredepl>val</inredepl>

<last>val</last>

</response>

**JSON Response**

"response":{"inredepl":"val","last":"val"}

| Parameter | Description |
| --- | --- |
| inredepl | Value is true or false.  true indicates that redeployment is in progress and connectivity is not available.  false indicates that redeployment is not running and all connectivity is available |
| last | Timestamp of last redeployment |

# 4 Customer APIs

The API service allows you to create and delete customer entries and to retrieve a list or details of customer entries. You can use a customer entry to link the entry to a scenario package deployment. The link enables monitoring from the customer perspective.

| Mode | Description |
| --- | --- |
| mode=add | Add a customer record |
| mode=del | Delete one or more customer records |
| mode=getList | Get a list of available customer records |
| mode=get | Get detailed information of a customer |

## 4.1 Adding Customers

Create one or multiple customer entries in the integration framework. The API adds all correct entries and skips incorrect entries. You can find detailed information in the response document.

method=cust

mode=add

**XML Request**

<request>

<entry>

<name>val</name>

<street>val</street>

<city>val</city>

<country>val</country>

<phone>val</phone>

<email>val</email>

<cat>val</cat>

<attachment>

<bfa:io pltype="xxx">any base64 string</bfa:io>

</attachment>

</entry>

…

</request>

**JSON Request**

"request": {

"arr": [

{"name":"val", "street":"val", "city":"val", "country":"val", "phone":"val", "email": "val"", "cat": "val"},

…

]

}

| Parameter | Description |
| --- | --- |
| name | Mandatory field to define the customer name |
| street | Optional field to define the street and house number |
| city | Mandatory field to define the zip code and city |
| country | Mandatory field to define the country |
| phone | Optional field to define the phone number |
| email | Optional field to define the e-mail address |
| cat | Optional field to define the category, T=Test, P=Productive |
| attachment | Optional binary file in framework XML representation (<bfa:io>). The data must be in base64 format. This is not supported in JSON |
| pltype | Mandatory payload type for the attachment, for example, jpg, bpm, and so on |

**XML Response**

<response>

<entry id="val" status="val" desc="val"/>

…

</response>

**JSON Response**

"response": {

"arr": [

{ "id":"val", "status":"val", "desc":"val" },

…

]

}

}

| Parameter | Description |
| --- | --- |
| id | Customer name |
| status | Entry status. 3.000 for success, otherwise an error number. For more information, see the appendix |
| desc | Long text of entry status |

## 4.2 Deleting Customers

Delete one or multiple customer records in the integration framework. The API service deletes all correct entries and skips incorrect entries. For more information, see the response document. You can delete a customer record by providing the name or the technical identifier.

method=cust

mode=del

**XML Request**

<request>

<entry>

<name>val</name>

</entry>

<entry>

<id>val</id>

</entry>

…

</request>

**JSON Request**

"request": {

"arr": [ {"name":"val"}, {"id":"val"}, … ]

}

| Parameter | Description |
| --- | --- |
| name | Customer name |
| id | Technical customer identifier, alternative option to delete a customer |

Note that either of the two parameters is mandatory.

**XML Response**

<response>

<entry id="val" name="val" status="val" desc="val"/>

…

</response>

**JSON Response**

"response": {

"arr": [

{ "id":"val", "name":"val", "status":"val", "desc":"val" },

…

]

}

}

| Parameter | Description |
| --- | --- |
| id | Technical identifier of customer entry |
| name | Customer name |
| status | Entry status. 3.000 for success, otherwise an error number. For more information, see the appendix |
| desc | Long text of entry status |

## 4.3 Getting a Customer List

Get a list of all customer entries in the integration framework.

method=cust

mode=getlist

**Request**

A request document is not required.

**XML Response**

<response>

<customer name="val" id="val"/>

…

</response>

**JSON Response**

"response": {"arr": [

{ "name":"val", "id":"val" },

…

]

}

| Parameter | Description |
| --- | --- |
| name | Customer name |
| id | Technical identifier of customer entry |

## 4.4 Getting Customer Details

Get details for a customer entry, either by using the customer name or the technical identifier.

method=cust

mode=get

**XML Request**

<request>

<id>val</id>

</request>

<request>

<name>val</name>

</request>

**JSON Request**

"request": {"id":"val"}

"request": {"name":"val"}

| Parameter | Description |
| --- | --- |
| name | Customer name |
| id | Technical identifier of customer entry |

**XML Response**

<response>

<id>val</id>

<name>val</name>

<street>val</street>

<city>val</city>

<country>val</country>

<phone>val</phone>

<email>val</email>

<cat>val</cat>

<att>val</att>

<associated qmsg="val" qerr="val">

<deployment ipac="val" name="val" id="val" status="val" qmsg="val"   
 qerr="val">

<qstep id="val" scen="val" q="val" s="val" cnt="val"   
 ipostatus="val" itcnt="val" exc.txt="val"   
 exc.ts="val" exc.tid="val"/>

...

</deployment>

...

</associated>

</response>

**JSON Response**

"response": {

"id":"val",

"name":"val",

"street":"val",

"city":"val",

"country":"val",

"phone":"val",

"email":"val",

"cat":"val",

"att":"val",

"associated": {

"qmsg":"val", "qerr":"val",

"deployment": [

{ "ipac":"val",

"name":"val",

"id":"val",

"status":"val",

"qmsg":"val",

"qerr":"val",

"qstep": [

{ "id":"val",

"scen":"val",

"q":"val",

"s":"val",

"cnt":"val",

"ipostatus":"val",

"itcnt":"val",

"exc.txt":"val",

"exc.ts":"val",

"exc.tid":"val"

},

...

]

},

...

]

}

}

| Parameter | Description |
| --- | --- |
| id | Technical identifier of customer entry |
| name | Customer name |
| street | Street and house number |
| city | Zip code and city |
| country | Country |
| phone | Phone number |
| email | E-mail address |
| cat | Optional category (T=Test, P=Productive) |
| att | Optional name of attachment, for example, 19030414324227812545C0A8B27C0F8B.png |
| associated  qmsg  qerr | Array of scenario package deployments linked to the customer.  Number of messages in queue for the customer.  Number of queue-triggered scenario steps with errors for the customer. |
| deployment  ipac  name  id  status  qmsg  qerr | Scenario package deployments linked to the customer.  Scenario package identifier  Deployment identifier  Technical deployment identifier  Deployment status (active or inactive)  Number of messages in queue for this scenario package deployment.  Number of queue-triggered scenario steps with errors of this scenario package deployment. |
| qstep  id  scen  q  s  cnt  ipostatus  itcnt  exc.txt  exc.ts  exc.tid | Scenario step triggered by queue  Name of scenario step  Name of associated scenario  Name of the queue to trigger this scenario step  Name of the queue-stream to trigger this scenario step  Total number of messages in the queue for this scenario step  Status of the scenario step ("1,error", "2,active", "3,inactive", "4,not activated")  Number of performed retrials in case of an error  Reported exception information in case of an error  Timestamp of last error  Transaction ID of last processing in case of an error. Allows you to get transaction-related information by calling the API (method=trans, mode=getinfo) |

## 4.5 Getting Customer Attachments

Get an attachment that is linked to the customer entry. You can get the attachment by name, by the technical customer identifier or by the technical attachment identifier.

method=cust

mode=getatt

**XML Request**

<request>

<id>val</id>

</request>

<request>

<name>val</name>

</request>

<request>

<att>val</att>

</request>

**JSON Request**

"request": {"id":"val"}

"request": {"name":"val"}

"request": {"att":"val"}

| Parameter | Description |
| --- | --- |
| name | Customer name |
| id | Technical identifier of customer entry |
| att | Name of attachment document |

**XML Response**

<bfa:io pltype="b64">any base64 string</bfa:io>

You can only get the response in XML format, not in JSON.

## 4.6 Updating Customers

Update customer entries.

method=cust

mode=modify

**XML Request**

<request xmlns:bfa="urn:com.sap.b1i.bizprocessor:bizatoms" >

<name>val</name> or <id>val</id> to identify the customer entry

<street>val</street>

<city>val</city>

<country>val</country>

<phone>val</phone>

<email>val</email>

<cat>val</cat>

<attachment>

<bfa:io pltype="xxx">any base64 string</bfa:io>

</attachment>

</request>

**JSON Request**

"request": {"name":"val", "id":"val", "street":"val", "city":"val", "country":"val", "phone":"val", "email": "val", "cat": "val"}

| Parameter | Description |
| --- | --- |
| name | Customer name, option to identify the customer that is going to be updated |
| id | Technical identifier of customer entry, alternative option to identify the customer that is going to be updated |
| street | Optional field to update the street and house number |
| city | Optional field to update the zip code and city |
| country | Optional field to update the country |
| phone | Optional field to update the phone number |
| email | Optional field to update the e-mail address |
| cat | Optional field to update the category, T=Test, P=Productive |
| attachment | Optional binary file in framework XML representation (<bfa:io>). The data must be in base64 format. Not supported by JSON |

**XML Response**

<response></response>

**JSON Response**

"response": {}

# 5 Activation APIs

The API service allows you to activate or to deactivate a deployment of a scenario package.

| Deployment Options | Description |
| --- | --- |
| mode=activate | Activate a deployment for a scenario package |
| mode=deactivate | Deactivate a deployment for a scenario package |

## 5.1 Activating a Deployment

Activate a scenario package deployment. The deployment must be valid. To validate the deployment, see section Validating a Deployment

Depending on the size of scenario steps that must be activated, the service can run some seconds.

method=act

mode=activate

**Request**

A request document is not required. The ipac and deplid URL parameters are mandatory.

**Response**

There is no response document.

## 5.2 Deactivating a Deployment

Deactivate a deployment of a scenario package.

method=act

mode=deactivate

**Request**

A request document is not required. The ipac and deplid URL parameters are mandatory.

**Response**

There is no response document.

## 5.3 Deactivating All Deployments

Deactivate all deployments of a scenario package.

method=act

mode=deactAll

**Request**

A request document is not required. The ipac URL parameter is mandatory.

**XML Response**

<response>

<deactDepl>val</deactDepl>

<remDocs>val</remDocs>

</response>

**JSON Response**

"response": {"deactDepl":"val", "remDocs":"val"}

| Parameter | Description |
| --- | --- |
| deactDepl | Number of deactivated deployments |
| remDocs | Number of removed documents (IPO documents, deployment documents, deployment definition documents) |

## 5.4 Unsubscribing Scenario Steps

Remove the subscription to an SAP Business One event or to an SAP ERP IDoc for scenario steps. Using the ipac and deplid URL parameters is mandatory.

method=act

mode=unsubscribe

**XML Request**

<request>

<scen>val</scen>

<step>val</step>

</request>

**JSON Request**

"request": {"scen":"val", "step":"val" }

| Parameter | Description |
| --- | --- |
| scen | Scenario identifier |
| step | Scenario step identifier |

**Response**

There is no response document.

## 5.5 Batch Mode for Scenario Package Activation

Activate multiple scenarios in one batch. The method consolidates all changes and triggers redeployment exactly once.

method=act

mode=activateBatch

**XML Request**

<request>

<entry ipac="val" deplid="val"/>

...

</request>

**JSON Request**

"request": {"arr":[{"ipac":"val","deplid":"val"},...]}

| Parameter | Description |
| --- | --- |
| ipac | Scenario package identifier |
| deplid | Deployment identifier. If empty, the action applies to all deployments. |

**XML Response**

<response>

<entry ipac="val" deplid="val" action="val" status="val" info="val" />

...

</response>

**JSON Response**

"response": {"batch":[{"ipac":"val","deplid":"val","action":"val","status":"val","info":"val"},...]}

| Parameter | Description |
| --- | --- |
| ipac | Scenario package identifier |
| deplid | Deployment identifier |
| action | activate triggers activation  no does not trigger activation, either because the scenario package or deployment do not exist, or the scenario package or deployment are already active |
| status | Entry-specific status number, 00 indicates that activation is triggered.  00 Activation triggered  01 Scenario package identifier is empty  02 Deployment identifier is empty  03 Scenario Package does not exist  04 Deployment does not exist  05 Deployment is already active  06 n.a.  07 Deployment is not valid |
| info | Entry-specific status description |

## 5.6 Batch Mode for Scenario Package Deactivation

Deactivate multiple scenarios in one batch. The method consolidates all changes and triggers redeployment exactly once.

method=act

mode=deactivateBatch

**XML Request**

<request>

<entry ipac="val" deplid="val"/>

...

</request>

**JSON Request**

"request": {"arr":[{"ipac":"val","deplid":"val"},...]}

| Parameter | Description |
| --- | --- |
| ipac | Scenario package identifier |
| deplid | Deployment identifier. If empty, the action applies to all deployments. |

**XML Response**

<response>

<entry ipac="val" deplid="val" action="val" status="val" info="val" />

...

</response>

**JSON Response**

"response": {"batch":[{"ipac":"val","deplid":"val","action":"val","status":"val","info":"val"},...]}

| Parameter | Description |
| --- | --- |
| ipac | Scenario package identifier |
| deplid | Deployment identifier |
| action | deactivate triggers deactivation  no does not trigger deactivation, because the scenario package or deployment do not exist or the scenario package or deployment is already inactive |
| status | Entry-specific status number, 00 indicates that activation is triggered.  00 Activation triggered  01 Scenario package identifier is empty  02 Deployment identifier is empty  03 Scenario Package does not exist  04 Deployment does not exist  05 Deployment is already active  06 n.a.  07 Deployment is not valid |
| info | Entry-specific status description |

## 5.7 Getting Status Information for Scenario Package Deployments

Get the status of scenario package deployments. The request document allows you to define a list of packages and deployments. If you do not provide a request document, the API returns a list of all packages and deployment in your integration framework.

method=act

mode=getStatus

**XML Request**

<request>

<entry ipac="val" deplid="val"/>

...

</request>

**JSON Request**

"request": {"arr":[{"ipac":"val","deplid":"val"},...]}

| Parameter | Description |
| --- | --- |
| ipac | Scenario package identifier |
| deplid | Deployment identifier |

**XML Response**

<response>

<entry ipac="val" deplid="val" status="val" info="val" />

...

</response>

**JSON Response**

"response": {"batch":[{"ipac":"val","deplid":"val","status":"val","info":"val"},...]}

| Parameter | Description |
| --- | --- |
| ipac | Scenario package identifier |
| deplid | Deployment identifier |
|  |  |
| status | Entry specific status number.   1. Scenario Package identifier is empty 2. Deployment identifier is empty 3. Scenario Package doesn't exist 4. Deployment doesn't exist 5. Deployment is not valid 6. Deployment is active 7. Deployment is inactive |
| info | Entry-specific status description |

# 6 System Landscape Directory (SLD) APIs

Since the SLD is a shared component, you can also use the SLD APIs for the integration framework version 1.

## 6.1 Getting SLD Entries

Get a list of SLD entries, either the complete list or define a filter for the list.

You can set a filter by SLD category, by system type identifier, by system group, or by an adapter identifier.

* Framework users can define SLD categories that group SLD systems.
* The following system type identifiers are available:

| System Type | Description |
| --- | --- |
| B1.2004 | SAP Business One 2004 and minor releases |
| B1.2005 | SAP Business One 2005 and minor releases |
| B1.2007 | SAP Business One 2007 and minor releases |
| B1.8.8 | SAP Business One 8.8 and minor releases |
| B1.9.0 | SAP Business One 9.0 and minor releases |
| BI.3.5.3 | SAP Business Warehouse 3.5.3 |
| BI.7.0.3 | SAP Business Warehouse 7.0.3 |
| E.AnySystem | E-Mail Server |
| ECC6.0 | SAP ERP Central Component, 6.0 |
| F.AnySystem | File System |
| F.B1.2004 | File System, inbound in format of SAP Business One 2004 |
| F.B1.2005 | File System, inbound in format of SAP Business One 2005 |
| F.B1.2007 | File System, inbound in format of SAP Business One 2007 |
| H.AnySystem | HTTP Access Active and Passive |
| J.AnySystem | JDBC Access to Database |
| P.AnySystem | FTP Server |
| R3.46C | SAP ERP System 3.46 C |
| R3.47.100 | SAP ERP System 3.47 100 |
| R3.47.200 | SAP ERP System 3.47 200 |
| T.AnySystem | Scheduler Configuration |
| W.AnySystem | Web Service Access Active and Passive |

* The following system groups are available:

| System Group | Description |
| --- | --- |
| B1 | SAP Business One system |
| BI | SAP Business (Warehouse) Intelligence System |
| DB | Systems with database connectivity (SAP Business One, J.AnySystem) |
| EMAIL | E-Mail system |
| ERP | SAP ERP systems |
| FILE | File systems |
| FTP | FTP servers |
| HTTP | HTTP systems |
| TIMER | Scehduler configuration, timer systems |
| WS | Web Services |

* The following adapters have SLD assignments:

| Adapters | Description |
| --- | --- |
| B1AO.1 | SAP Business One Outbound - asynchronous |
| B1EV | SAP Business One Inbound by Events |
| B1SL | SAP Business One Service Layer Call |
| B1SO | SAP Business One Outbound via Service Layer |
| B1II | SAP Business One Inbound via HTTP |
| B1IO | SAP Business One Outbound, including key mapping |
| B1DI | SAP Business One DI API Call |
| BYDI | SAP Business ByDesign Inbound |
| BYDO | SAP Business ByDesign Outbound, including key mapping |
| CRON | Timer Trigger |
| FILE | Store File to File System |
| FILI | File Inbound |
| FILP | Get File from File System (Polling) |
| FILR | File System Directory Information |
| FILO | File Outbound |
| FTPO | Store File to FTP Server |
| FTPP | Get File from FTP Server (Polling) |
| HTTA | HTTP - Outgoing Call |
| HTTP | HTTP - Incoming Call |
| IDOC | SAP ERP - Incoming IDoc |
| JDBC | Database SQL Call |
| MAIR | Receive E-Mail |
| MQIN | Multiple Queue Inbound |
| RFCA | SAP ERP - RFC Call |
| RFCP | SAP ERP - Incoming RFC Call |
| SMTE | Send Email Call |
| SMTP | Send Email Outbound |
| WSAN | Web Service - Outgoing Notification |
| WSAO | Web Service - Incoming One-Way Call |
| WSAR | Web Service - Incoming Request/Response Call |
| WSAS | Web Service - Outgoing Solicit/Response Call |

method=sld

mode=getlist

**XML Request**

* SLD category filter definition:

<request> <category>val</category> </request>

* System type identifier filter definition:

<request><systype>val</systype></request>

* System group filter definition:

<request> <sysgroup>val</sysgroup> </request>

* Adapter identifier filter definition:

<request> <adapter>val</adapter> </request>

**JSON Request**

* SLD category filter definition:

"request": { "category":"val" }

* System type identifier filter definition:

"request": { "systype":" val" }

* System group filter definition:

"request": { "sysgroup":"val" }

* Adapter identifier filter definition:

"request": { "adapter":"val" }

**XML Response**

<response>

<list filter="val">

<entry sysid="val" systype="val" name="val"/>

…

</list>

</response>

**JSON Response**

"response": {

"filter": "val",

"list": [

{ "sysid":"val", "systype":"val", "name":"val" },

…

]

}

| Parameter | Description |
| --- | --- |
| filter | Information about the used filter in the form of filtertype=value, for example, adapter='B1'.  If the filter is empty, there was no definition by systype, sysgroup, adapter or category. |
| sysid | System identifier of SLD entry, for example, 0010000102 |
| systype | System type identifier, for example, J.AnySystem |
| name | SLD entry name |

## 6.2 Testing Connections

Test the connection to a system defined in SLD. Depending on the adapter type, the service can take some time.

method=sld

mode=tstcon

**XML Request**

<request>

<sysid>val</sysid>

</request>

**JSON Request**

"request": {"sysid":"val", "adapter":"val"}

| Parameter | Description |
| --- | --- |
| sysid | System identifier of SLD entry, for example, 0010000102 |
| adapter | Adapter identifier, HTTA, B1SL, RFCA, JDBC, FILI, FILO, WSAN, WSAS, MAIR, SMTP, FTPP, FTPO, CRON |

**XML Response**

<response>

<sysid>val</sysid>

<adapter>val</adapter>

<test.status>val</test.status>

<test.info>val</test.info>

</response>

**JSON Response**

"response": {

"sysid": "val",

"adapter": "val",

"test.status": "val",

"test.info": "Connect to SBODemoDE(JDBC) successfully"

}

| Parameter | Description |
| --- | --- |
| sysid | System identifier of SLD entry, for example, 0010000102 |
| adapter | Adapter identifier, HTTA, B1SL, RFCA, JDBC, FILI, FILO, WSAN, WSAS, MAIR, SMTP, FTPP, FTPO, CRON |
| test.status | Status of connection test, value can be success or failure |
| test.info | Status message, in case of failure, the connection test returns the original error message from the API |

## 6.3 Adding SLD Entries

Add an SLD entry.

method=sld

mode=add

**XML Request**

<request>

<systype>val</systype>

<sysid>val</sysid>

<crypt>val</crypt>

<name>val</name>

<cid>val</cid>

<cat>val</cat>

<parameter>

…

</parameter>

</request>

**JSON Request**

"request": {

"systype":"val", "sysid":"val", "crypt":"val", "name":"val",   
 "cid":"val", "cat":"val", "parameter": { … }

}

| Parameter | Description |
| --- | --- |
| systype | Mandatory parameter to define the system type.  Find the available system types in the table of the Retrieving a List of SLD Entries section |
| sysid | Optional parameter to define the system identifier of the new entry.  If the value is empty, the API generates a number.  Provide a ten digits number using the 001xxxXXXX syntax.   * xxx is the vendor namespace, the unique vendor abbreviation. * XXXX is consecutive numbering starting with 0000. |
| crypt | Optional parameter, either true or false.  If set to true, the API encrypts all passwords. This is the default if the parameter is not handed over. If set to false, the API stores all passwords in plain text. |
| name | Optional parameter defining the name of the SLD entry. If empty, the API sets the name to the value of the generated sysid. |
| cid | Optional parameter to define a custom identifier |
| cat | Optional name of SLD category. If the category is not available, the service creates it |
| parameter | Optional section to define the values of connectivity properties.  To define a property using the parameter, use the name of the label of the SLD entry.  If you do not define the connectivity property value, the API uses the default value. Find the property name and default values in the tables of the appendix  For FTP systems, the values are the same for both connectivity types FTPO and FTPP. |

**XML Response**

<response>

<op>add</op>

<systype>val</systype>

<sysid>val</sysid>

<crypt>val</crypt>

<name>val</name>

<cid>val</cid>

<cat>val</cat>

<par.x internal="val">val</val>

...

</response>

**JSON Response**

"response": {

"op": "add",

"systype": "val",

"sysid": "val",

"crypt": "val",

"name": "val",

"cid":"val",

"cat":"val",

"par.x": { "par":"val", "value":"val", "internal":"val" },

…

}

| Parameter | Description |
| --- | --- |
| op | Operation add |
| systype | System type |
| sysid | System identifier generated by the API, or defined by the API consumer |
| crypt | Value of parameter crypt handed over to the call |
| name | SLD entry name, either handed over by the API consumer or generated by the API |
| cid | Optional parameter to define a custom identifier |
| cat | Optional name of SLD category |
| par.x | For each parameter, the API returns the parameter name, the value and the parameter number of the internal API |

## 6.4 Updating SLD Entries

Change parameters of an SLD entry. You cannot change the system type (systype).

method=sld

mode=upd

**XML Request**

<request>

<sysid>val</sysid>

<crypt>val</crypt>

<name>val</name>

<cid>val</cid>

<cat>val</cat>

<parameter>

…

</parameter>

</request>

**JSON Request**

"request": {

"sysid":"val", "crypt":"val", "name":"val","cid":"val","cat":"val",

"parameter": { … }

}

| Parameter | Description |
| --- | --- |
| sysid | Mandatory parameter to define the system identifier of the SLD entry you want to update |
| crypt | Optional parameter (true, false).   * If set to true, the API encrypts all passwords. This is the default, if you do not hand over a value. * If set to false, the API stores all passwords in plain text. |
| name | Optional parameter, defining SLD entry name |
| cid | Optional parameter to define a custom identifier |
| cat | Optional name of SLD category. If a category is already assigned, the entry moves to the new category. The service deletes the old category, if there are no other entries assigned. To unassign the category, set cat=unassign |
| parameter | Optional section to define connectivity properties values.  To define a property using parameter, use the name of the label of the SLD entry. For more information, refer to the previous section,  If you do not define a connectivity property, the API uses the existing value. |

**XML Response**

<response>

<op>upd</op>

<systype>val</systype>

<sysid>val</sysid>

<crypt>val</crypt>

<name>val</name>

<cid>val</cid>

<cat>val</cat>

<par.x internal="val">val</val>

...

</response>

**JSON Response**

"response": {

"op": "add",

"systype": "val",

"sysid": "val",

"crypt": "val",

"name": "val",

"cid":"val",

"cat":"val",

"par.x": { "par":"val", "value":"val", "internal":"val" },

…

}

| Parameter | Description |
| --- | --- |
| op | Operation upd |
| systype | System type |
| sysid | System identifier |
| crypt | Value of crypt parameter handed over to the call |
| name | SLD entry name handed over to the call |
| cid | Optional custom identifier |
| cat | Optional name of SLD category |

## 6.5 Deleting SLD Entries

Delete an SLD entry. The service deletes an assigned category, if no other system is assigned to the category.

method=sld

mode=del

**XML Request**

<request>

<sysid>val</sysid>

</request>

**JSON Request**

"request": { "sysid":"val" }

| Parameter | Description |
| --- | --- |
| sysid | Mandatory parameter to define the system identifier of the entry you want to delete |

**Response**

There is no response document.

## 6.6 Copying SLD Entries

Copy an SLD entry in SLD. The name of the copied system is the name of the source system followed by .copy.

* When copying an FTP system (P.AnySystem), the service copies the parameters of the FTPO connectivity section to both, the FTPO and the FTPP section.
* When copying a system of type ECC6.0 or W.AnySystem, the service copies the parameter values of the WSAN connectivity section to both the WSAN and WSAS connectivity sections.
* For SAP Business One system types, the service changes the company parameter to company.copy and the company name in the url parameter in the JDBC section to <company>.copy.

method=sld

mode=cpy

**XML Request**

<request>

<from>val</from>

<to>val</to>

</request>

**JSON Request**

"request": { "from":"val", "to":"val" }

| Parameter | Description |
| --- | --- |
| from | Mandatory parameter to define the source system identifier |
| to | Optional parameter to define the system identifier of the entry that the API generates.  If the value is empty, the API generates a new number.  Provide a ten digits number using the 001xxxXXXX syntax.   * xxx is the vendor namespace, the unique vendor abbreviation. * XXXX is consecutive numbering starting with 0000. |

**XML Response**

<response>

<op>cpy</op>

<systype>val</systype>

<sysid>val</sysid>

<crypt>val</crypt>

<name>val</name>

<cid>val</cid>

<cat>val</cat>

<par.x internal="val">val</par.x>

...

</response>

**JSON Response**

"response": {

"op": "add",

"systype": "val",

"sysid": "val",

"crypt": "val",

"name": "val",

"cid":"val",

"cat":"val",

"par.x": { "par":"val", "value":"val", "internal":"val" },

…

}

| Parameter | Description |
| --- | --- |
| op | Operation cpy |
| systype | System type |
| sysid | System identifier, either generated by the API or defined by the API consumer |
| crypt | false |
| name | SLD entry name with .copy added |
| cid | Optional custom identifier |
| cat | Optional SLD category |
| par.x | For each parameter, the API returns the parameter name, the value and the parameter number of the internal API |

## 6.7 Getting SLD Entry Details

Get the parameters of an SLD entry.

method=sld

mode=getdetail

**XML Request**

<request>

<sysid>val</sysid>

</request>

**JSON Request**

"request": { "sysid":"val" }

| Parameter | Description |
| --- | --- |
| sysid | System identifier of SLD entry to request detailed information |

**XML Response**

<response>

<sysid>val</sysid>

<name>val</name>

<cid>val</cid>

<cat>val</cat>

<systype>val</systype>

<ConnectivityTypes>

<ConnectivityType id="val">

<par>val</par>

...

</ConnectivityType>

...

</ConnectivityTypes>

</response>

**JSON Response**

"response": {

"sysid": "val",

"name": "val",

"cid":"val",

"cat":"val",

"systype": "val",

"ConnectivityTypes": [

{ "id": "val", "<par>":"val", … }

]

}

| Parameter | Description |
| --- | --- |
| sysid | System identifier of SLD entry |
| name | SLD entry name |
| cid | Optional custom identifier |
| cat | Optional SLD category |
| systype | System type of SLD entry |
| ConnectivityTypes | Array of connectivity types |
| ConnectivityType | Connectivity type with all parameters |
| id | Connectivity type identifier |
| <par> | Parameter and value |

## 6.8 Getting System Types and Connectivity Types

Get a list of available system types and their connectivity types from SLD.

method=sld

mode=getstypes

**Request**

A request document is not required.

**XML Response**

<response>

<systype id="val" name="val">

<contype>val</contype>

...

</systype>

...

</response>

**JSON Response**

"response": {

"systypes": [

{ "id":"val", "name":"val", "ConnectivityList": ["val", …] },

…

]

}

| Parameter | Description |
| --- | --- |
| id | System type identifier of SLD entry |
| name | SLD entry name |
| systype | System type of SLD entry |
| contype | Connectivity type associated to the system type. In JSON result array ConnectivityList |

## 6.9 Getting System Type Properties

Get a list of connectivity type parameters that are linked to the system type.

method=sld

mode=getstpar

**XML Request**

<request>

<systype>val</systype>

</request>

**JSON Request**

"request": { "systype":"val" }

| Parameter | Description |
| --- | --- |
| systype | System type identifier |

**XML Response**

<response>

<systype id="val" name="val">

<contype id="val" name="val">

<par name="val">

<enum key="val" value="val" desc="val"/>

...

</par>

...

</contype>

...

</systype>

</response>

**JSON Response**

"response": {

"id": "val",

"name": "val",

"ConnectivityList": [

{ "id":"val", "name":"val",

"Parameters": [

{"id": "val", enum: [ {"key":"val", "value":"val", "desc":"val"}, … ] },

…

]

{

]

}

| Parameter | Description |
| --- | --- |
| id | System type identifier |
| name | System type name |
| ConnectivityList id | Connectivity type identifier |
| ConnectivityList name | Connectivity type name |
| parameter id | Parameter identifier |
| enum | Optional enumeration for parameter |
| key | Enumeration identifier |
| value | Enumeration value |
| desc | Enumeration description |

## 6.10 Getting SLD Categories

Retrieve a list if SLD categories including assigned SLD entries.

method=sld

mode=getcatlist

**Request**

A request document is not required.

**XML Response**

<response>

<category name="val" cnt="val">

<entry id="val" name="val"/>

...

</category>

...

</response>

**JSON Response**

"response": {

"categories": [

{"name":"val","cnt":"val","SysIdList":[{"id":"val","name":"val"},…]},…

]

}

| Parameter | Description |
| --- | --- |
| name | Category name |
| cnt | Number of SLD entries in category |
| entry/SysIdList id | SLD entry SysId |
| entry/SysIdList name | SLD entry name |

## 6.11 Creating an SLD Category

Create an SLD category.

method=sld

mode=addcat

**XML Request**

<request>

<cat>val</cat>

</request>

**JSON Request**

"request": {"cat":"val"}

| Parameter | Description |
| --- | --- |
| cat | Category name |

**Response**

There is no response document.

## 6.12 Deleting an SLD Category

Delete an SLD category. Deleting categories with SLD entries is not possible.

method=sld

mode=delcat

**XML Request**

<request>

<cat>val</cat>

</request>

**JSON Request**

"request": {"cat":"val"}

| Parameter | Description |
| --- | --- |
| cat | Category name |

**Response**

There is no response document.

## 6.13 Setting Categories for SLD Entries

Set categories for SLD entries. You can define multiple categories and associate SLD entries. If the category is not yet available, the service creates the category. The API processes valid system identifiers (SysIds) with length of 10. Category sections must be unique and a system identifier can be associated to exactly one category.

method=sld

mode=setcat

**XML Request**

<request>

<cat id="val" mode="val">

<sysid>val</sysid>

...

</cat>

...

</request>

**JSON Request**

"request": [{"id":"val","mode":"val","sysid":["val", "val", ...]},...]

| Parameter | Description |
| --- | --- |
| id | Category name |
| mode | The following modes are available:   * Use 'add' to add SysId entries to the category. This is the default * Use 'replace' to replace entries in the category with defined SysId entries.   Available entries are removed from the category. |
| sysid | SLD System identifier |

**XML Response**

<response>

<entry cat="val" sysid="val" status="val" action="val"/>

...

</response>

**JSON Response**

"response": {"arr":[{"cat":"val","sysid":"val","status":"val","action":"val"},...]}

| Parameter | Description |
| --- | --- |
| cat | Category name |
| sysid | SLD System identifier |
| status | 'ok' or 'error - sysid does not exist in SLD' |
| action | The performed action.   * 'no' if no action is taken due to an error * 'remain' if the entry is already assigned to the category * 'new' if the entry is newly assigned to the category * 'remove' if the entry is removed from the category |

# 7 Transaction APIs

## 7.1 Getting Transaction Information

Get detailed information about a transaction processed by the integration framework.

method=trans

mode=getinfo

**XML Request**

<request>

<tid>val</tid>

</request>

**JSON Request**

"request": {"tid":"val"}

| Parameter | Description |
| --- | --- |
| tid | Transaction identifier (32-byte GUID) |

**XML Response**

<response>

<timeStamp>val</timeStamp>

<trstatus>val</trstatus>

<duration>val</duration>

<inMsgSize>val</inMsgSize>

<maxMsgSize>val</maxMsgSize>

<type>val</type>

…

<!—Additional information for B1i 2.0 transactions -->

<ipac>val</ipac>

<deplid>val</deplid>

<scen>val</scen>

<step>val</step>

<sysid>val</sysid>

<!—Additional information for B1i 1.0 transactions -->

<sysid>val</sysid>

<ipo>val</ipo>

<step.type>val</step.type>

<system>val</system>

<phase>val</phase>

</response>

**JSON Response**

"response": {

"timeStamp": "val",

"trstatus": "val",

"duration": "val",

"inMsgSize": "val",

"maxMsgSize": "val",

"type": "val",

…

<!—additional information for B1i 1.0 transactions -->

"sysid": "val",

"ipo": "val",

"step.type": "val",

"system": "val",

"phase": "val"

<!—additional information for B1i 2.0 transactions -->

"ipac": "val",

"deplid": "val",

"scen": "val",

"step": "val"

"sysid": "val"

}

| Parameter | Description |
| --- | --- |
| timestamp | Start timestamp of transaction processing |
| trstatus | Transaction status:   * STARTED: Transaction is started and still in processing * INCOMMIT: Transaction is in second (commit) phase * INROLLBACK: Transaction is in rollback phase due to an exception * COMPLETED: Transaction finished successfully * CANCELLED: Transaction is canceled due to an exception |
| duration | Duration in milliseconds |
| inMsgSize | Size of the incoming message in byte |
| maxMsgSize | Maximal message size during processing |
| type | Usage type of transaction:   * B1i Ver.1: integration framework version 1 * B1i Ver.2: integration framework version 2 * SLD Service: xxxx * iService: xxxx (integration service of version 2) * B1 Event Dispatcher: xxxx * B1iP Service: xxxx (integration platform service) * B1i API Service Ver.1 * B1i API Service Ver.2 * B1i FW Service: xxxx (integration framework version 1 services) * User Service: xxxx |
| sysid | Framework version 1: system identifier the transaction is associated to |
| ipo | Framework version 1: transaction type identifier   * INB\_AY\_TIMR\_ASYN\_BIU * INB\_B1\_EVNT\_ASYN\_EVT * INB\_DB\_TIMR\_ASYN\_FIX * INB\_DB\_TIMR\_DATARETRIEVER * INB\_FI\_EXST\_ASYN\_FIX * INB\_FI\_EXST\_ASYN\_FSL * INB\_FI\_EXST\_ASYN\_FSL * INB\_FI\_EXST\_ASYN\_NAM * INB\_FI\_EXST\_ASYN\_XPT * INB\_HT\_CALL\_SYNC\_XPT * INB\_IQ\_INTQ\_ASYN\_BIU * INB\_IQ\_INTQ\_ASYN\_QS * INB\_R3\_IDOC\_ASYN\_XPT * INB\_WS\_CALL\_SYNC\_XPT * OUT\_B1 * OUT\_BP * OUT\_DB * OUT\_FILE * OUT\_HT * OUT\_R3 * OUT\_WS * PRC\_AY * PRC\_B1 * PRC\_FIf * PRC\_FIf * PRC\_FIi * PRC\_Fin * PRC\_Fix * PRC\_QS * PRC\_R3 * PRQ\_AY * PRQ\_B1 * PRQ\_FIf * PRQ\_FIi * PRQ\_Fin * PRQ\_Fix * PRQ\_IQ * PRQ\_QS * PRQ\_R3 * VOID |
| step.type | Framework version 1: Scenario step or related error handler |
| system | Framework version 1: Defines sender and receiver system type:   * BP (business process) * Database * Internal Queue * Flat File * HTTP Call * SAP Business One * SAP ERP * Timer * Void * Web Service Call |
| phase | Framework version 1: Defines the phase:   * inbound * processing * outbound |
| ipac | Framework version 2: Scenario package identifier |
| deplid | Framework version 2: Deployment identifier |
| scen | Framework version 2: Scenario identifier |
| step | Framework version 2: Scenario step identifier |
| sysid | Framework version 2: System identifier, if deployment is linked to an SLD entry |

# 8 Scenario Package Configuration APIs

## 8.1 Getting the Scenario Package Configuration

Get detailed information about the scenario package configuration.

method=ipac

mode=getcfg

## 8.2 Deleting a Scenario Package

Deletes a scenario package and related artefacts. There are several artefacts that are associated to a scenario package. Define the artefacts that you want to delete.

Archive the scenario package before using the deletion mode.

method=ipac

mode=del

**XML Request**

<request>

<design>val</design>

<setup>val</setup>

<test>val</test>

<archive>val</archive>

<bpm>val</bpm>

<mappingmeta>val</mappingmeta>

<addds>val</addds>

</request>

**JSON Request**

"request": {

"design":"val",

"setup":"val",

"test":"val",

"archive":"val",

"bpm":"val",

"mappingmeta":"val",

"addds":"val"

}

| Parameter | Description |
| --- | --- |
| design | Set value to true to delete all design time documents of the scenario package, including all scenario step definitions and configurations, mapping definitions, generated mapping XSL documents and scripts, all XSL and JavaScript documents and all test documents. Default value is false. |
| setup | Set value to true to delete the setup. All deployments are deactivated. All setup definitions, generated flows and transactions are deleted. Subscriptions to SAP Business One events and/or to IDocs of SAP ERP are unsubscribed. The related deployment document in the system.xc group is removed. Default value is false |
| test | Set value to true to delete the test environment, including setups, all test documents and related documents in the system.xc group. Default value is false. |
| archive | Set value to true to delete all archive versions of the scenario package. Default value is false. |
| bpm | Set value to true to delete an optionally available BPM model. Default value is false. |
| mappingmeta | Set value to true to delete all metadata for mapping projects, including inbound schema definitions, outbound schema definitions, schema validation XSDs and test messages.  Before using the parameter make sure that the documents are not used by other scenario packages. Default value is false. |
| addds | Set value true to delete all associated additional datasets.  Before using the parameter make sure that the documents are not used by other scenario packages. Default value is false. |

**XML Response**

<response>

<unsubscribed>val</unsubscribed>

<removed.docs>val</removed.docs>

<remove.design>val</remove.design>

<remove.setup>val</remove.setup>

<remove.deploydoc>val</remove.deploydoc>

<remove.unsubscribe>val</remove.unsubscribe>

<remove.test>val</remove.test>

<remove.archive>val</remove.archive>

<remove.bpm>val</remove.bpm>

<remove.mappingmeta>val</remove.mappingmeta>

<remove.addds>val</remove.addds>

</response>

**JSON Response**

"response": {

"unsubscribed": "val",

"removed.docs": "val",

"remove.design": "val",

"remove.setup": "val",

"remove.deploydoc": "val",

"remove.unsubscribe": "val",

"remove.test": "val",

"remove.archive": "val",

"remove.bpm": "val",

"remove.mappingmeta": "val",

"remove.addds": "val"

}

| Parameter | Description |
| --- | --- |
| unsubscribed | Number of unsubscriptions for SAP Business One events and/or SAP ERP IDocs |
| removed.docs | Total number of deleted documents |
| remove.design | API call setting for design parameter |
| remove.setup | API call setting for setup parameter |
| remove.deploydoc | API call setting for setup parameter |
| remove.unsubscribe | API call setting for setup parameter |
| remove.test | API call setting for test parameter |
| remove.archive | API call setting for archive parameter |
| remove.bpm | API call setting for bpm parameter |
| remove.mappingmeta | API call setting for mappingmeta parameter |

# 9 Internal Queue APIs

## 9.1 Getting the Number of Messages in a Queue

Get the number of available messages in a queue and stream. If the stream is not defined, the service returns the total number and a list of all available streams with the number of messages per stream.

method=queue

mode=getcnt

**XML Request**

<request>

<q>val</q>

<s>val</s>

</request>

**JSON Request**

"request": {"q":"val", "s":"val"}

| Parameter | Description |
| --- | --- |
| q | Internal queue name. This parameter is mandatory |
| s | Optional name of internal stream |

**XML Response**

<response>

<q>val</q>

<s>val</s>

<cnt>val</cnt>

<streamlist>

<stream name="val" cnt="val"/>

...

</streamlist>

</response>

**JSON Response**

"response": {

"q":"val",

"s":"val",

"cnt": "val",

"streamlist": {

"name": "val",

"cnt": "val"

}, …

}

| Parameter | Description |
| --- | --- |
| q | Internal queue name. This parameter is mandatory |
| s | Optional name of internal stream |
| cnt | Number of messages. If stream is not defined, the value is the total number of messages in all streams of the requested queue, otherwise the number of messages in the requested queue/stream. |
| streamlist | Optional and if stream is undefined, a list of all streams with the number of messages. |

## 9.2 Getting Information About Messages in the Queue

Retrieve the number of messages in a queue and stream. Define the queues by values based on scenario packages. The service only considers queues and streams that are related to scenario packages.

method=queue

mode=getcntscen

**XML Request**

<request>

<ipac>value</ipac>

<scen>value</scen>

<step>value</step>

<depl>value</depl>

<sysid>value</sysid>

<obj>value</obj>

<custid>value</custid>

<adptr>value</adptr>

<dmode>value</dmode>

</request>

**JSON Request**

"request": {

"ipac":"val",

"scen":"val",

"step":"val",

"depl":"val",

"sysid":"val",

"obj":"val",

"custid":"val",

"adptr":"val",

"dmode":"val"

}

| Parameter | Description |
| --- | --- |
| ipac | Optional scenario package name |
| scen | Optional scenario name, ipac parameter is mandatory |
| step | Optional scenario step name, ipac and scen parameters are mandatory |
| depl | Optional deployment identifier, ipac parameters is mandatory |
| sysid | Optional sender system. Scenario steps triggered by B1EV, IDOC or MQIN support multiple sender systems. Set a filter to the sender system |
| obj | Optional business object, ipac parameter is mandatory.  Define the business object value by using the Q-Info property. Inbound adapter technically mapped to queue inbound allow you to define the property. You can define a fixed value. The following inbound adapter support Q-Info: DBQI, ENQI, B1EV, IDOC, BPMI, BYDI and MQIN. |
| custid | Optional customer identifier |
| adptr | Optional inbound adapter. Inbound adapters are DBQI, ENQI, B1EV, IDOC, BPMI, BYDI and MQIN |
| dmode | Optional parameter to define to retrieve information for all queues (dmode=all) or only for queues with messages (dmode=msg). Default value is all. |

**XML Response**

<response>

<filter ipac="val" scen="val" step="val" depl="val" sysid="val" obj="val"  
 custid="val" adptr="val" dmode="val"/>

<total cnt="0"/>

<qlist>

<entry ipac="val" scen="val" step="val" depl="val" cnt="val"   
 custid="val" custn="val" sysid="val" obj="val" q="val" s="val"   
 ipostatus="val" adptr="val" itCnt="val" exc.txt="val"   
 exc.ts="val" exc.tid="val"/>

...

</qlist>

</response>

**JSON Response**

"response": {

"filter": {

"ipac": "val",

"scen": "val",

"step": "val",

"depl": "val",

"sysid": "val",

"obj": "val",

"custid": "val",

"adptr": "val",

"dmode": "val"

},

"total": {"cnt":"val"},

"qlist": [

{

"ipac": "val",

"scen": "val",

"step": "val",

"depl": "val",

"cnt": "val",

"custid": "val",

"custn": "val",

"sysid": "val",

"obj": "val",

"q": "val",

"s": "val",

"ipostatus": "val",

"adptr": "val",

"itCnt": "val",

"ext.txt": "val",

"ext.ts": "val",

"ext.tid": "val"

},

…

]

}

| Parameter | Description |
| --- | --- |
| filter | Echoes inbound parameter |
| total / cnt | number of messages in the queue based on defined filter criteria. |
| qlist | List of queue/streams matching the filter criteria |
| ipac | Name of scenario package related to queue/stream entry |
| scen | Name of scenario related to queue/stream entry |
| step | Name of scenario step related to queue/stream entry |
| depl | Name of deployment identifier related to queue/stream entry |
| cnt | Number of messages in queue/stream |
| custid | Customer identifier related to queue/stream entry |
| custn | Name and short address of customer related to queue/stream entry |
| sysid | SLD System identifier of sender system related to this queue/stream entry |
| obj | Business object related to queue/stream entry |
| q | Name of the internal queue related to queue/stream entry |
| s | Name of stream related to queue/stream entry |
| ipostatus | Status of transaction (IPO) triggered by queue/stream entry.  Status can be active (IPO is running, status ok), inactive (IPO is deactivated in deployment panel), not activated (IPO does not exist), error (IPO currently inactive, last processing ended in exception). |
| adptr | Inbound adapter that triggered the scenario step related to queue/stream entry |
| itCnt | For queue/stream entries where last processing ended in exception: Number of attempts to process the message again |
| ext.txt | For queue/stream entries where last processing ended in exception: Error text of exception |
| ext.ts | For queue/stream entries where last processing ended in exception: Timestamp |
| ext.tid | For queue/stream entries where last processing ended in exception: Transaction identifier |

## 9.3 Getting Messages with IDs and Trace IDs from Queue

Get messages from a queue/stream. The service returns a message list with message and trace identifier.

method=queue

mode=getlist

**XML Request**

<request>

<q>val</q>

<s>val</s>

<limit>val</limit>

<prev>val</prev>

<traceid>val</traceid>

</request>

**JSON Request**

"request": {"q":"val","s":"val","limit":"val","prev":"val","traceid":"val"}

| Parameter | Description |
| --- | --- |
| q | Mandatory internal queue name |
| s | Mandatory name of internal stream |
| limit | Optional parameter to limit the list for reading block by block |
| prev | Optional parameter to set the last read msgid to continue with next black for block read. |
| traceid | Optional parameter to set a filter to queue entries with traceid |

**XML Response**

<response>

<q>val</q>

<s>val</s>

<limit>val</limit>

<traceid>val</traceid>

<prev>val</prev>

<list>

<entry msgid="val" traceid="val"/>

...

</msglist>

</response>

**JSON Response**

"response": {

"q": "val",

"s": "val",

"limit": "val",

"traceid": "val",

"prev": "val",

"list": [ { "msgid": "21", "traceid": "" }, … ]

}

| Parameter | Description |
| --- | --- |
| q | Internal queue name |
| s | Name of internal stream |
| limit | Limit for reading block by block |
| prev | Last read msgid |
| traceid | traceid filter value |
| list  msgid  …traceid | Message number of queue message  Trace ID value of queue message |

## 9.4 Getting a Message from a Queue

Get a message from the queue. The message is **not** consumed. The method is only available for XML payload.

method=queue

mode=getmsg

**XML Request**

<request>

<q>val</q>

<s>val</s>

<msgid>val</msgid>

</request>

| Parameter | Description |
| --- | --- |
| q | Mandatory internal queue name |
| s | Mandatory name of internal stream |
| msgid | Mandatory message identifier |

**XML Response**

<response>

<msg q="val" s="val" msgid="val">

message

</msg>

</response>

| Parameter | Description |
| --- | --- |
| q | Internal queue name |
| s | Name of internal stream |
| msgid | Message identifier |

## 9.5 Deleting a Message from a Queue

Delete a message from the queue.

method=queue

mode=delmsg

**XML Request**

<request>

<q>val</q>

<s>val</s>

<msgid>val</msgid>

</request>

**JSON Request**

"request":{ "q":"val", "s":"val", "msgid":"val" }

| Parameter | Description |
| --- | --- |
| q | Mandatory internal queue name |
| s | Mandatory name of internal stream |
| msgid | Mandatory message identifier |

**Response**

There is no response document.

## 9.6 Clearing Queues

Deletes all messages of a queue. You can filter messages by traceid. The service triggers the deletion. The deletion process runs asynchronously in background.

method=queue

mode=clear

**XML Request**

<request>

<q>val</q>

<s>val</s>

<traceid>val</traceid>

</request>

**JSON Request**

"request":{ "q":"val", "s":"val", "traceid":"val" }

| Parameter | Description |
| --- | --- |
| q | Mandatory internal queue name |
| s | Mandatory name of internal stream |
| traceid | Optional trace ID filter value |

**Response**

There is no response document.

# 10 User Administration APIs

## 10.1 Getting a User List

Get a list of administration or runtime users.

method=users

mode=list

**XML Request**

<request>

<admin>val</admin>

<runtime>val</runtime>

</request>

**JSON Request**

"request": {"admin":"val", "runtime":"val"}

| Parameter | Description |
| --- | --- |
| admin | Set the value to true to get a list of administration users. Default value is false |
| runtime | Set the value to true to get list of runtime users. Default value is false |

**XML Response**

<response>

<admin.users.list>

<user name="val" roles="val" lang="val" active="val" expires="val"/>

...

</admin.users.list>

<runtime.users.list>

<user name="val" lang="val" active="val" expires="val"/>

...

</runtime.users.list>

</response>

**JSON Response**

"response": {

"admin.users.list": [

{ "name":"val", "roles":"val", "lang":"val", "active":"val", "expires":"val" },

...

],

"runtime.users.list": [

{ "name":"val", "lang":"val", "active":"val", "expires":"val" },

...

]

}

| Parameter | Description |
| --- | --- |
| admin.users.list | List of administration users |
| runtime.users.list | List of runtime users |
| name | User name |
| roles | Space-separated list of roles that are associated to the user. Relevant only for administration user. The role can be Administrator, Developer, Configurator or Supervisor. |
| lang | Language that is associated to the user |
| active | Indicates if the user account is active or locked |
| expires | Expiration date of user entry |

## 10.2 Creating Users

Create an administration or runtime user.

method=users

mode=add

**XML Request**

<request>

<type>val</type>

<usr>val</usr>

<pwd>val</pwd>

<roles>val</roles>

<lang>val</lang>

<active>val</active>

<email>val</email>

<name>val</name>

</request>

**JSON Request**

"request": {"type":"val", "usr":"val", "pwd":"val", "roles":"val", "lang":"val", "active":"val", "email":"val", "name":"val"}

| Parameter | Description |
| --- | --- |
| type | The mandatory parameter defines the user type, either admin or runtime |
| usr | The mandatory parameter defines the unique user name |
| pwd | The mandatory parameter defines the initial password of the user account. During first login, the user is required to change the initial password. |
| roles | The mandatory parameter defines one or multiple roles. Separate the roles by blank. The role can be Administrator, Developer, Configurator or Supervisor. Not relevant for runtime users. |
| lang | The optional parameter sets the language. Default is en |
| active | The optional parameter activates or deactivates the user account. Default is true. |
| email | Optional parameter to define the e-mail used to send new password. Not relevant for runtime users. |
| name | Optional parameter to define a full name used to send new password. Not relevant for runtime users. |

**XML Response**

<response>

<mode>api.add</mode>

<defDoc>val</defDoc>

<usr>val</usr>

<pwd>val</pwd>

<roles>val</roles>

<lang>val</lang>

<active>val</active>

<email>val</email>

<name>val</name>

</response>

**JSON Response**

"response": {"mode":"api.add", "defDoc":"val", "usr":"val", "pwd":"val", "roles":"val", "lang":"val", "active":"val", "email":"val", "name":"val"}

| Parameter | Description |
| --- | --- |
| mode | Fixe value api.add |
| defDoc | Name of the changed authentication document. For admin, defDoc is users\_bfd.xml. For runtime, defDoc is users\_ipo.xml. |
| usr | User name |
| pwd | Initial password of user account. |
| roles | User roles |
| lang | Default language of user. |
| active | Indicates if user account is active. |
| email | Registered e-mail of user. |
| name | Registered name of user. |

## 10.3 Deleting Users

Delete an administration or runtime user.

method=users

mode=del

**XML Request**

<request>

<type>val</type>

<usr>val</usr>

</request>

**JSON Request**

"request": {"type":"val", "usr":"val"}

| Parameter | Description |
| --- | --- |
| type | Mandatory parameter that defines the user type., either admin or runtime. |
| usr | Mandatory user name |

**XML Response**

<response>

<mode>api.del</mode>

<defDoc>val</defDoc>

<usr>val</usr>

</response>

**JSON Response**

"response": {"mode":"api.del", "defDoc":"val", "usr":"val"}

| Parameter | Description |
| --- | --- |
| mode | Fixed value api.del |
| defDoc | Name of the changed authentication document. For admin, defDoc is users\_bfd.xml. For runtime, defDoc is users\_ipo.xml. |
| usr | User name |

# 11 Cockpit APIs

## 11.1 Getting Cockpit Results

Retrieve cockpit information of tiles for the control center, the framework version 1 and 2.

method=cpit

mode=run

**XML Request**

<request>

<type>val</type>

<id>val</id>

</request>

**JSON Request**

"request": {"type":"val", "id":"val"}

| **Parameter** | **Description** |
| --- | --- |
| type | Define the cockpit:   * cc from control center cockpit * 1 tiles of the framework version 1 cockpit * 2 tiles of the framework version 2cockpit. |
| id | Tile identifier |

Tile identifier of the control center cockpit:

* Last Errors in IPO Steps
* Deployed IPO Steps
* Stopped Steps Due to Errors
* Stopped Steps Due to Recovery
* Manually Stopped Steps
* Stopped Steps Due to Deployment
* Completed Executions
* InCommit Executions
* Cancelled Executions
* InRollback Executions
* DB Queues
* DBQ Streams
* DBQ Messages
* BizStore Datasets
* BizStore Groups
* BizStore Documents
* Doc Cache Accesses
* Doc Cache Hit Rate
* App Init Errors
* App Init Warnings
* Deployed B1i Apps
* Maximum Memory
* Used Memory
* Detected CPU Cores
* Scheduler Threads
* Last Start of B1iP
* Crashed Last Time?

Tile identifiers of framework version 1 cockpit:

* MsgLog – Success
* MsgLog – Failed
* MsgLog – Filtered
* MsgLog – InProcess
* Error Inbox: Receiver Systems with Error
* Error Inbox: Technical Errors
* Blocked Queues
* Canceled Transactions
* Scenario Transactions
* Scenario Packages (total)
* non-SAP Scenario Packages (total)
* Scenario Packages (active)
* non-SAP Scenario Packages (active)
* Scenario Processes (active)
* Scenario Processes (deactivated)
* Activation Conflicts
* Namespace Conflicts
* Database Size
* BizStore Documents
* Used Memory (Tomcat)
* Used Memory
* Used CPUs

Tile identifiers of framework version 1 cockpit:

* Scenario Packages
* Deployments
* Activated Deployments
* Deployed Scenario Steps
* Activated Scenario Steps
* Messages in Error Inbox
* DB Queues
* DBQ Streams
* DBQ Messages
* Database Size
* BizStore Documents
* Used Memory
* Used CPUs

**XML Response**

<response>

<href>val</href>

<tt>val</tt>

<res>val</res>

<col>val</col>

<icon>val</icon>

<icol>val</icol>

</response>

**JSON Response**

"response":  
  {"href":"val","tt":"val","res":"val","col":"val","icon":"val","icol":"val"}

| **Parameter** | **Description** |
| --- | --- |
| href | Link (BizStore URI) to related user interface |
| tt | Tooltip, short description of tile. |
| res | Result |
| col | Color of result value (color name or HTML color code) |
| icon | Name of UI5 icon |
| icol | Color of alert icon (color name or HTML color code) |

# 12 Transaction Protocol APIs

## 12.1 Getting Aggregated Values for a Day, Month or Year

Get the duration, number of processed transactions and size of inbound messages of transactions for a day, a month or year.

method=trprtc

mode=getentry

**XML Request**

<request>

<day>val</day>

<month>val</month>

<year>val</year>

<status>val</status>

<pid>val</pid>

<tid>val</tid>

<pac>val</pac>

<scen>val</scen>

<step>val</step>

<depl>val</depl>

<snd>val</snd>

<cid>val</cid>

</request>

**JSON Request**

"request": {

"day":"val",

"month":"val",

"year":"val",

"status":"val",

"pid":"val",

"tid":"val",

"pac":"val",

"scen":"val",

"step":"val",

"depl":"val",

"snd":"val",

"cid":"val"

}

| **Parameter** | **Description** |
| --- | --- |
| day | Optional two-digit entry for day |
| month | Optional two-digit entry for month |
| year | Optional two-digit entry for year |
| status | Optional transaction status. Default is all transactions.  0 transaction with STARTED status  1 transaction with INCOMMIT status  2 transaction with INROLLBACK status  3 transaction with COMPLETED status  4 transaction with CANCELLED status |
| pid | Optional partner identifier. Only relevant for reports in iPaaS controller |
| tid | Optional tenant identifier (iPaaS client identifier). Only relevant for reports in iPaaS controller |
| pac | Optional scenario package identifier |
| scen | Optional scenario |
| step | Optional scenario step |
| depl | Optional deployment identifier |
| snd | Optional sender system (SLD SysId) |
| cid | Optional customer identifier |

**XML Response**

<response>

<duration>val</duration>

<count>val</count>

<size>val</size>

</response>

**JSON Response**

"response": {"duration":"val","count":"val","size":"val"}

| **Parameter** | **Description** |
| --- | --- |
| duration | Aggregated duration time in ms |
| count | Aggregated number of processed messages |
| size | Aggregated size of inbound messages in bytes |

## 12.2 Getting Series of Aggregated Values for a Period

Retrieve a series of durations, number of processed transactions and sizes of inbound messages. The response document lists days, month or years.

method=trprtc

mode=getperiod

**XML Request**

<request>

<from>

<day>val</day>

<month>val</month>

<year>val</year>

</from>

<to>

<day>val</day>

<month>val</month>

<year>val</year>

</to>

<dim>val</dim>

<res>val</res>

<status>val</status>

<pid>val</pid>

<tid>val</tid>

<pac>val</pac>

<scen>val</scen>

<step>val</step>

<depl>val</depl>

<snd>val</snd>

<cid>val</cid>

</request>

**JSON Request**

"request": {

"from":{"day":"val","month":"val","year":"val"},

"to":{"day":"val","month":"val","year":"val"},

"dim":"val",

"res":"val",

"status":"val",

"pid":"val",

"tid":"val",

"pac":"val",

"scen":"val",

"step":"val",

"depl":"val",

"snd":"val",

"cid":"val"

}

| Parameter | Description |
| --- | --- |
| from/day | Mandatory two-digit start day |
| from/month | Mandatory two-digit start month |
| from/year | Mandatory two-digit start year |
| to/day | Mandatory two-digit end day |
| to/month | Mandatory two-digit end month |
| to/year | Mandatory two-digit end year |
| dim | (day, month, year) Defines to retrieve the list by days, by month or by year |
| res | Reduce retrieved data by defining the result. Set res (res=C, res=S, res=D) to retrieve only count, size or duration information |
| status | Optional transaction status. Default is all transactions.  0 transaction with STARTED status  1 transaction with INCOMMIT status  2 transactions with INROLLBACK status  3 transaction with COMPLETED status  4 transaction with CANCELLED status |
| pid | Optional partner identifier. Only relevant for reports in iPaaS controller |
| tid | Optional tenant identifier (iPaaS client identifier). Only relevant for reports in iPaaS controller |
| pac | Optional scenario package identifier |
| scen | Optional scenario |
| step | Optional scenario step |
| depl | Optional deployment identifier |
| snd | Optional sender system (SLD SysId) |
| cid | Optional customer identifier |

**XML Response**

<response>

<entry>

<ts>val</ts>

<duration>val</duration>

<count>val</count>

<size>val</size>

</entry>

...

</response>

**JSON Response**

"response": {

"period": [

{"ts":"val", "duration":"val", "count":"val", "size":"val"},

...

]

}

| Parameter | Description |
| --- | --- |
| ts | If dim=day, timestamp in YYMMDD format  If dim=month, timestamp in YYMM format  If dim=year, timestamp in YY format |
| duration | Aggregated duration in ms |
| count | Aggregated number of processed messages |
| size | Aggregated size of inbound messages in bytes |

# Appendix

## 1 Status Messages

### 1.1 General Messages

| ID | Description |
| --- | --- |
| 000 | ok |
| 0.001 | Mandatory 'method' parameter is empty |
| 0.002 | Mandatory 'ipac' parameter is empty |
| 0.003 | Mandatory 'deplid' parameter is empty |
| 0.004 | 'xxxx' method unknown |
| 0.005 | Mode of 'xxxx' method is empty |
| 0.006 | 'xxxx' mode of 'xxxx' method unknown |
| 0.007 | 'xxxx' scenario package does not exist |
| 0.008 | Request structure missing |

### 1.2 Messages of Deployment-Specific Properties Service

| ID | Description | Mode |
| --- | --- | --- |
| 1.000 | Ok | all modes |
| x entries added | add |
| x entries deleted | del |
| x entries set | set |
| 1.001 | Identifier is empty | add, del, set |
| 1.002 | Deployment-specific property already exists | add |
| 1.003 | Duplicate identifier | add, set |
| 1.004 | Identifier cannot start with a number | add |
| 1.005 | Identifier cannot start with 'x' | add |
| 1.006 | Identifier cannot start with 'xml' | add |
| 1.007 | Identifier (xxxx) contains wrong characters at position x | add |
| 1.008 | Deployment-specific property does not exist, deletion not possible | del |
| 1.009 | x errors occurred, x entries added | add |
| 1.010 | x errors occurred, x entries deleted | del |
| 1.011 | Deployment-specific property does not exist | set |
| 1.012 | x errors occurred, x entries set | set |
| 1.013 | Deployment-specific properties for 'xxxx' do not exist | del, set |

### 1.3 Deployment Service Messages

| ID | Description | Mode |
| --- | --- | --- |
| 2.000 | ok | all modes |
| 2.001 | Framework runs in single tenancy mode. Deployment already exists, cannot add deployment | add |
| 2.002 | 'xxxx' deployment identifier already exists | add |
| 2.003 | Deployment identifier contains wrong characters at position x | add |
| 2.004 | Customer identifier does not exist | add |
| 2.005 | 'xxxx' deployment identifier does not exist | del, set |
| 2.006 | Deployment with x scenario steps is active, deletion not possible | del, set |
| 2.007 | Deployment already exists, and mode is not set to 'override' | set |
| 2.008 | Request structure is missing | set, setfilter |
| 2.009 | Scenario package does not have any scenarios assigned | set |
| 2.010 | Scenario package does not have scenarios steps assigned | set |
| 2.011 | 'xxxx' scenario does not exist in package | set, setfilter, delfilter |
| 2.012 | 'xxxx' scenario step does not exist in package | set, setfilter, delfilter |
| 2.013 | 'xxxx' scenario step does not exist in 'xxxx' scenario of scenario package | set, setfilter, delfilter |
| 2.014 | 'xxxx' port does not exist | set |
| 2.015 | No sysid entries defined for port | set |
| 2.016 | Invalid 'xxxx' sysid entry assigned to 'xxxx' port. Length not equal 10 | set |
| 2.017 | Invalid 'xxxx' sysid entry assigned to port. ID does not start with '001' | set |
| 2.018 | Invalid 'xxxx' sysid entry assigned to 'xxxx' port. ID does not exist | set |
| 2.019 | Invalid 'xxxx' sysid assigned to 'xxxx' port. 'xxxx' systype does not match 'xxxx' adapter | set |
| 2.020 | Only 1 sysid assignment allowed to 'xxxx' port ('xxxx' adapter) | set |
| 2.021 | No scenario definition | setfilter |
| 2.022 | No scenario step definition | setfilter |
| 2.023 | No rules definition | setfilter |
| 2.024 | Rule type has wrong value (‘sender’, ‘receiver’) | setfilter, delfilter |
| 2.025 | Rule definition wrong – sender system is empty | setfilter, delfilter |
| 2.026 | Rule definition wrong – receiver system is empty | setfilter, delfilter |
| 2.027 | Sender system does not exist | setfilter, delfilter |
| 2.028 | Receiver system does not exist | setfilter, delfilter |
| 2.029 | Rule definition wrong – line definition missing | setfilter |
| 2.030 | Line definition wrong - 'andor' parameter is empty | setfilter |
| 2.031 | Line definition wrong - 'andor' parameter has wrong value (AND, OR) | setfilter |
| 2.032 | Line definition wrong - 'crit' parameter is empty | setfilter |
| 2.033 | Line definition wrong (crit='xxx') - criteria field is not defined | setfilter |
| 2.034 | Line definition wrong (crit='xxx') - selection xpath is empty in criteria field definition | setfilter |
| 2.035 | Line definition wrong - 'op' paramter has wrong value (equal to, not equal to, greater than, less than, between, not between, greater than or equal to, less than or equal to, is in, contains, equal to blank, is not blank) | setfilter |
| 2.036 | Rule type 'xxx' does not fit to criteria field definition with type='xxx' | setfilter |
| 2.037 | Port is not unique | set |

### 1.4 Customer Service Messages

| ID | Description | Mode |
| --- | --- | --- |
| 3.000 | Ok | modify |
| x entries added | add |
| x entries deleted | del |
| 3.001 | Name is empty | add |
| 3.002 | 'xxxx' customer already exists | add |
| 3.003 | City is empty | add |
| 3.004 | Country is empty | add |
| 3.005 | x errors occurred, x entries added | add |
| 3.006 | Parameter name and id empty | del, att, modify |
| 3.007 | Customer does not exist | del, att, modify |
| 3.008 | x errors occurred, x entries deleted | del |
| 3.009 | There is no attachment for the customer entry |  |
| 3.010 | Attachment identifier is empty |  |
| 3.011 | Attachment does not exist | att |
| 3.012 | Parameter 'cat' has wrong value; allowed 'P' and 'T' | add |
| 3.013 | Attachment is not in bfa:io format | add |
| 3.014 | Payload type for attachment is not defined | add |
| 3.015 | Customer with this name exists already | modify |

### 1.5 Activation Service Messages

| ID | Description | Mode |
| --- | --- | --- |
| 4.000 | OK | all modes |
| 4.001 | 'xxxx' deployment identifier does not exist | activate, deactivate, unsubscribe |
| 4.002 | Deployment is active already | activate |
| 4.003 | Deployment is not valid | activate |
| 4.004 | Activation rejected | activate |
| 4.005 | Deployment is not set up | activate |
| 4.006 | No scenario or scenario steps selected for activation | activate |
| 4.007 | Deployment is already inactive | deactivate |
| 4.008 | Deactivation rejected | deactivate |
| 4.009 | Scenario does not exist | unsubscribe |
| 4.010 | Scenario step does not exist | unsubscribe |
| 4.011 | Parameter 'scen' is empty | unsubscribe |
| 4.012 | Parameter 'step' is empty | unsubscribe |
| 4.013 | Scenario step trigger is not event-based | unsubscribe |
| 4.014 | No subscription available | unsubscribe |
| 4.015 | No entries in batch | activateBatch, deactivateBatch |
| 4.016 | No valid entries in batch | activateBatch, deactivateBatch |

### 1.6 SLD Service Messages

| ID | Description | Mode |
| --- | --- | --- |
| 5.000 | OK | all modes |
| 5.001 | 'xxxx' system type unknown | getlist, add, getstpar |
| 5.002 | 'xxxx' system group unknown | getlist |
| 5.003 | 'xxxx' adapter unknown | getlist |
| 5.004 | 'xxxx' category unknown | getlist |
| 5.005 | SysId is empty | tstcon, upd, del, cpy, getdetail |
| 5.006 | Adapter is empty | tstcon |
| 5.007 | 'xxxx' SysId does not exist in SLD | tstcon, upd, del, cpy, getdetail |
| 5.008 | SysId has wrong format | tstcon, upd, del, cpy, getdetail |
| 5.009 | 'xxxx' SysId does not support 'xxxx' adapter | tstcon |
| 5.010 | ‘xxxx’ adapter unknown | tstcon |
| 5.011 | System type is empty | add, getstpar |
| 5.012 | ‘xxxx’ SysId already exists | add, upd |
| 5.013 | Invalid 'xxxx' value for 'xxxx' parameter | add, upd |
| 5.014 | 'xxxx' name already exists | add, upd, cpy |
| 5.015 | Error message from internal API | add, upd, del, cpy |
| 5.016 | Source SysId in 'from' parameter is empty | cpy |
| 5.017 | Source SysId in 'from' parameter has invalid format | cpy |
| 5.018 | Source ‘xxxx’ SysId in 'from' parameter does not exist in SLD | cpy |
| 5.019 | Destination SysId in ‘to’ parameter has invalid format | cpy |
| 5.020 | Destination ‘xxxx’ SysId in 'to' parameter already exists in SLD | cpy |
| 5.021 | Parameter ‘cat‘ is empty | addcat, delcat |
| 5.022 | Category exists already | addcat |
| 5.023 | Category does not exist | delcat |
| 5.024 | SysIds are linked to category, deletion not allowed | delcat |
| 5.025 | There is no category definition in request document | setcat |
| 5.026 | There is no category definition with valid identifier in request document | setcat |
| 5.027 | There is no category with valid sysid definitions in request document | setcat |
| 5.028 | Category definitions are not unique | setcat |
| 5.029 | Sysid(s) associated to multiple categories | setcat |
| 5.030 | Sysid duplicated in category section | setcat |

### 1.7 Transaction Service Messages

| ID | Description | Mode |
| --- | --- | --- |
| 7.000 | Ok | getinfo |
| 7.001 | Transaction ID is empty | getinfo |
| 7.002 | No information available for transaction ID | getinfo |

### 1.8 Scenario Packages Service Messages

| ID | Description | Mode |
| --- | --- | --- |
| 6.000 | Ok |  |
| 6.001 | Scenario package does not exist |  |

### 1.9 Internal Queue Service Messages

| ID | Description | Mode |
| --- | --- | --- |
| 9.000 | ok | all modes |
| 9.001 | Queue is not defined | getcnt, getlist, getmsg, delmsg, clear |
| 9.002 | Stream is not defined | getlist, getmsg, delmsg, clear |
| 9.003 | Message identifier is not defined | getmsg, delmsg |
| 9.004 | This method is only available for XML | getmsg |
| 9.005 | Scenario Package does not exist | getcntscen |
| 9.006 | Filter on scenario, but scenario package not defined | getcntscen |
| 9.007 | Scenario does not exist | getcntscen |
| 9.008 | Filter on scenario step, but scenario package not defined | getcntscen |
| 9.009 | Filter on scenario step, but scenario not defined | getcntscen |
| 9.010 | Scenario step does not exist | getcntscen |
| 9.011 | Filter on deployment identifier, but scenario package not defined | getcntscen |
| 9.012 | Deployment identifier does not exist | getcntscen |
| 9.013 | SysId does not exist in SLD | getcntscen |
| 9.014 | Filter on business object, but scenario package not defined | getcntscen |
| 9.015 | Customer identifier does not exist | getcntscen |
| 9.016 | Inbound adapter definition is wrong | getcntscen |

### 1.10 Cockpit Service Messages

| ID | Description | Mode |
| --- | --- | --- |
| 10.000 | Ok | run |
| 10.001 | Parameter type is missing | run |
| 10.002 | Parameter id is missing | run |
| 10.003 | Parameter type is wrong (allowed values 1, 2, cc) | run |
| 10.004 | Parameter id does not exist in cockpit | run |
| 10.005 | Datasource not supported |  |

### 1.11 Transaction Protocol Service Messages

| ID | Description | Mode |
| --- | --- | --- |
| 11.000 | OK | getentry, getperiod |
| 11.001 | Time period (from,to) not correct | getentry |
| 11.002 | Parameter status has wrong value (0,1,2,3,4) | getentry |

### 1.12 User Service Messages

| ID | Description |
| --- | --- |
| A.000 | OK |
| A.001 | User type undefined |
| A.002 | Parameter type wrong |
| A.003 | User name missing |
| A.004 | User entry user name already exists |
| A.005 | Initial password missing |
| A.006 | Role definition missing |
| A.007 | Role definition wrong |

## 2 SLD System Type Properties and Default Values

### 2.1 Database System (J.AnySystem)

| Connectivity Type | Property | Default Value |
| --- | --- | --- |
| JDBC | driver | 'com.microsoft.sqlserver.jdbc.SQLServerDriver' |
| url | 'jdbc:sqlserver://IPADDRESS:1433;integratedSecurity=false;databaseName=DBNAME;selectMethod=cursor' |
| username | 'sa' |
| password | 'B1i' |

### 2.2 SAP Business One Systems (B1.2004, B1.2005, B1.2007, B1.8.8, B1.9.0)

| Connectivity Type | Property | Default Value |
| --- | --- | --- |
| B1DI | b1Server | '127.0.0.1' |
| licenseServer | '' |
| sldServer | '' |
| company | 'SBODemoUS' |
| dbType | '4' |
| dbUser | 'sa' |
| dbPassword | 'B1i' |
| userName | 'B1i' |
| password | 'B1i' |
| language | '3' |
| isTrust | 'false' |
| jcoPath | '' |
| diProxyhost | <b1Server> |
| diProxyport | '2099' |
| proxyHost | '' |
| proxyPort | '' |
| B1SL | b1Server | <b1Server> |
| company | <company> |
| destProtocol | 'http' |
| destHost | <b1Server> |
| destPort | '50001' |
| destPath | '/b1s/v1' |
| sluser | <userName> |
| slpassword | <password> |
| trustStoreURI | '' |
| JDBC | driver | 'com.microsoft.sqlserver.jdbc.SQLServerDriver' |
| url | 'jdbc:sqlserver://IPADDRESS:1433;integratedSecurity=false;databaseName=DBNAME;selectMethod=cursor' |
| username | 'sa' |
| password | 'B1i' |

### 2.3 SAP Business Warehouse Systems (BI Systems (BI.3.5.3, BI.7.0.3))

| Connectivity Type | Property | Default Value |
| --- | --- | --- |
| HTTA | destProtocol | 'http' |
| destHost | '127.0.0.1' |
| destPort | '' |
| destPath | '' |
| query | '' |
| proxyHost | '' |
| proxyPort | '' |
| method | 'post' |
| authentification | 'basic' |
| user | 'B1i' |
| password | 'B1i' |
| user2query | '' |
| password2query | '' |
| trustStoreURI | '' |
| keyStoreURI | '' |
| tlsVersion | '' |

### 2.4 E-Mail System (E.AnySystem)

| Connectivity Type | Property | Default Value |
| --- | --- | --- |
| SMTP | protocol | 'SMTP' |
| smtpHost | '127.0.0.1' |
| smtpPort | '25' |
| username | '' |
| password | '' |
| trustedHost | 'true' |
| MAIR | Protocol | 'POP3' |
| Host | '127.0.0.1' |
| Port | '110' |
| User | '' |
| Password | '' |
| UseSSL | 'true' |

### 2.5 HTTP System (H.AnySystem)

| Connectivity Type | Property | Default Value |
| --- | --- | --- |
| HTTA | destProtocol | 'http' |
| destHost | '127.0.0.1' |
| destPort | '' |
| destPath | '' |
| query | '' |
| proxyHost | '' |
| proxyPort | '' |
| method | 'post' |
| authentification | 'basic' |
| user | 'B1i' |
| password | 'B1i' |
| user2query | '' |
| password2query | '' |
| trustStoreURI | '' |
| keyStoreURI | '' |
| tlsVersion | '' |
| HTTP | associatedSrvIP | '' |

### 2.6 File Systems (F.AnySystem, F.B1.2004, F.B1.2005, F.B1.2007)

| Connectivity Type | Property | Default Value |
| --- | --- | --- |
| FILI | filePatternI | 'c:\b1if\in\\*.\*' |
| Encoding | '1' |
| Delimiter | ',' |
| WrapChar | ''' |
| PayloadType | '1' |
| RuleDoc | 'true' |
| FILO | filePatternO | 'c:\b1if\out\\*.xml' |

### 2.7 SAP ERP Systems, WS Sections Only for ECC 6.0

| Connectivity Type | Property | Default Value |
| --- | --- | --- |
| RFCA | applicationServer | '' |
| client | '' |
| user | '' |
| password | 'B1i' |
| language | 'EN' |
| systemNumber | '00' |
| maxConnections | '5' |
| gatewayHost | '' |
| gatewayServiceNumber | 'sapgw00' |
| senderPartner | '' |
| senderPort | '' |
| recieverPartner | '' |
| recieverPort | '' |
| SNC\_MODE | '' |
| SNC\_QOP | '' |
| SNC\_MYNAME | '' |
| SNC\_PARTNERNAME | '' |
| SNC\_LIB | '' |
| RFCP | applicationServer | <applicationServer> |
| client | <client> |
| user | <user> |
| password | <password> |
| language | <language> |
| systemNumber | <systemNumber> |
| maxConnections | <maxConnections> |
| gatewayHost | <gatewayHost> |
| gatewayServiceNumber | <gatewayServiceNumber> |
| programID | '' |
| unicode | 'true' |
| SNC\_MODE | <SNC\_MODE> |
| SNC\_QOP | <SNC\_QOP> |
| SNC\_MYNAME | <SNC\_MYNAME> |
| SNC\_PARTNERNAME | <SNC\_PARTNERNAME> |
| SNC\_LIB | <SNC\_LIB> |
| WSAS | destProtocol | 'http' |
| destHost | '' |
| destPort | '' |
| destPath | '' |
| query | '' |
| proxyHost | '' |
| proxyPort | '' |
| authentification | 'basic' |
| user | 'B1i' |
| password | 'B1i' |
| sslTruststorePath |  |
| sslTruststorePassword | '' |
| trustStoreURI |  |
| keyStoreURI | '' |
| WSAN | destProtocol | <destProtocol> |
| destHost | <destHost> |
| destPort | <destPort> |
| destPath | <destPath> |
| query | <query> |
| proxyHost | <proxyHost> |
| proxyPort | <proxyPort> |
| authentification | <authentification> |
| user | <user> |
| password | <password> |
| sslTruststorePath | <sslTruststorePath> |
| sslTruststorePassword | <sslTruststorePassword> |
| trustStoreURI | <trustStoreURI> |
| keyStoreURI | <keyStoreURI> |
| WSAR | associatedSrvIP | '' |

### 2.8 FTP System (P.AnySystem)

| Connectivity Type | Property | Default Value |
| --- | --- | --- |
| FTPO | protocol | 'FTP' |
| ftpServer | '127.0.0.1' |
| ftpPort | '21' |
| connectionMode | 'PASV' |
| userName | '' |
| password | '' |
| filePattern | '' |
| proxyHost | '' |
| proxyPort | '' |
| proxyUser | '' |
| proxyPassword | '' |
| proxyType | 'NOPROXY' |
| ssl\_trustStoreURI | '' |
| ssl\_trustStorePassword | '' |
| FTPP | protocol | <protocol> |
| ftpServer | <ftpServer> |
| ftpPort | <ftpPort> |
| connectionMode | <connectionMode> |
| userName | <userName> |
| password | <password> |
| filePattern | <filePattern> |
| proxyHost | <proxyHost> |
| proxyPort | <proxyPort> |
| proxyUser | <proxyUser> |
| proxyPassword | <proxyPassword> |
| proxyType | <proxyType> |
| ssl\_trustStoreURI | <ssl\_trustStoreURI> |
| ssl\_trustStorePassword | <ssl\_trustStorePassword> |

### 2.9 Timer System (T.AnySystem)

| Connectivity Type | Property | Default Value |
| --- | --- | --- |
| CRON | minute | '00' |
| hour | '\*' |
| day | '\*' |
| month | '\*' |
| dow | '\*' |
| year | '\*' |

# Copyrights, Trademarks, and Disclaimers

© Copyright 2020 SAP SE. All rights reserved.

The current version of the copyrights, trademarks, and disclaimers at <https://www.sap.com/about/legal/copyright.html> is valid for this document.