**Integration Framework**

**Schemas**

**Last Changed: August 5, 2018**

[1 vBIU Outbound Schemas 2](#_Toc521241092)

[1.1 Skipping Outbound Processing 2](#_Toc521241093)

[1.2 Multiple Single Messages Processing 2](#_Toc521241094)

[1.3 SAP Business One 4](#_Toc521241095)

[1.3.1 B1 Object Schema 4](#_Toc521241096)

[1.3.2 B1 Object (Full) Schema 5](#_Toc521241097)

[1.3.3 B1 Service Schema 6](#_Toc521241098)

[1.3.4 B1 Service (Full) Schema 7](#_Toc521241099)

[1.3.5 B1i SQL Schema 9](#_Toc521241100)

[1.3.6 B1 SQL Schema 11](#_Toc521241101)

[1.3.7 SQL (Full) Schema 12](#_Toc521241102)

[1.4 Database 13](#_Toc521241103)

[1.4.1 B1i SQL Database Schema 13](#_Toc521241104)

[1.4.2 SQL Database Schema 15](#_Toc521241105)

[1.4.3 SQL (Full) Database Schema 16](#_Toc521241106)

[1.5. Flat File 17](#_Toc521241107)

[1.5.1 XML Flat File Schema 17](#_Toc521241108)

[1.5.2 DSV Flat File Schema 17](#_Toc521241109)

[1.5.3 DSV (Full) Flat File Schema 19](#_Toc521241110)

[1.5.4 Text Flat File Schema 20](#_Toc521241111)

[1.5.5 Text (Full) Flat File Schema 22](#_Toc521241112)

[1.6 Web Service 24](#_Toc521241113)

[2. Process Atoms 25](#_Toc521241114)

[2.1 Atom Send E-Mail Schema 25](#_Toc521241115)

[2.2 Atom HTTP Call Schema 26](#_Toc521241116)

[2.3 Atom .NET Call Schema 27](#_Toc521241117)

[2.4 Correct after Branch 27](#_Toc521241118)

[2.5 Payload Type JSON 28](#_Toc521241119)

[3. Preconfiguration Schemas 29](#_Toc521241120)

[3.1 Scenario Package – Setup Description Schema 29](#_Toc521241121)

[Copyrights, Trademarks, and Disclaimers 31](#_Toc521241122)

# 1 vBIU Outbound Schemas

## 1.1 Skipping Outbound Processing

If you do not want to hand over the message to the receiver system, the scenario step can create a special tag that indicates to integration framework processing to skip the message processing. To skip outbound processing, use the <b1im\_skip> tag in the final transformation atom (atom0).

If message log creation is enabled, use the following attributes to control the creation of message log entries for skipped messages:

|  |  |
| --- | --- |
| **Attributes** | **Creation and Display of  Message Log** |
| <b1im\_skip/> | In Filtered section |
| <b1im\_skip msglog="true"/> | In Filtered section |
| <b1im\_skip msglog="false"/> | No message log |
| <b1im\_skip msgout="yes"/> | In Success section |
| <b1im\_skip msgout="yes" msglog="true"/> | In the Success section |
| <b1im\_skip msgout="yes" msglog="false"/> | No message log |
| <b1im\_skip failureout="yes"/> | In Failure section |
| <b1im\_skip failureout="yes" msglog="true"/> | In Failure section |
| <b1im\_skip failureout="yes" msglog="false"/> | No message log |
| <b1im\_skip msgout="yes" failureout="yes"/> | In Success section |
| <b1im\_skip msgout="yes" failureout="yes" msglog="true"/> | In Success section |
| <b1im\_skip msgout="yes" failureout="yes" msglog="false"/> | No message log |

Add the following to the final transformation atom to display an exception in the message log and in the error inbox, if a call or several calls in your processing have failed:

<b1im\_exception xmlns= ”” info=”my info” >

info

To display an individual message, define it using the info attribute.

For example: <b1im\_exception xmlns="" info="error message text"/>

In your process flow, set *Stop processing if fails* to false, and handle the error messages as well as the success messages at the outbound stage. You can add SAP Business One error messages of single or multiple calls to the outbound message of the message log.

## 1.2 Multiple Single Messages Processing

If you process multiple items, you usually want to hand them over to the receiver system as multiple single items. Indicate multiple single messaging with the root tag <b1im\_multimsg> and the tag <b1im\_msg> for each single message. The integration framework supports the tag for system types *SAP Business One*, *Database* and *File System*.

<!-- batch processing for multiple message outbound -->

<b1im\_multimsg>

<xsl:for-each select="$msg/… ">

<!-- processing for a single item -->

<b1im\_msg>

...

</b1im\_msg>

</xsl:for-each>

</b1im\_multimsg>

## 1.3 SAP Business One

### 1.3.1 B1 Object Schema

Example for business partner object

**<B1out type="object">**

<BusinessPartners xmlns="">

<row>

<CardCode>C20000</CardCode>

<CardName>Stehle GmbH</CardName>

<CardType>C</CardType>

<Address>My Home 25</Address>

<Phone1>49/63452632-323</Phone1>

<ContactPerson>Johann Meier</ContactPerson>

<EmailAddress>hugo.test@company.com</EmailAddress>

<Website>www.myc.de</Website>

</row>

<ContactEmployees>

<row>

<CardCode>C20000</CardCode>

<Name>Johann Meier</Name>

</row>

</ContactEmployees>

</BusinessPartners>

**</B1out>**

**Prerequisites**

You have defined the method, object identifier and object key in the outbound setup of the scenario step design, selecting *Scenarios* → *Step Design* → *[Outbound]*→ *[Details]*.

B1out/@type

This is the mandatory B1out root tag with the mandatory attribute with the value object.

### 1.3.2 B1 Object (Full) Schema

Example for business partner object

**<B1out type="object\_full">**

**<Control>**

<method>Update/Insert</method>

<objectid>2</objectid>

<keyname>CardCode</keyname>

**</Control>**

**<Payload>**

<BusinessPartners>

<row>

<CardCode>C20000</CardCode>

<CardName>Stehle GmbH</CardName>

<CardType>C</CardType>

<Address>My Home 25</Address>

<Phone1>49/63452632-323</Phone1>

<ContactPerson>Johann Meier</ContactPerson>

<EmailAddress>hugo.test@company.com

</EmailAddress>

<Website>www.myc.de</Website>

</row>

<ContactEmployees>

<row>

<CardCode>C20000</CardCode>

<Name>Johann Meier</Name>

</row>

</ContactEmployees>

</BusinessPartners>

**</Payload>**

**</B1out>**

B1out/@type

Mandatory B1out root tag, mandatory attribute with the value object\_full

Control/method

The following is available:

* Insert (for insert)
* Insert/Update (for insert with fallback to update)
* Update (for update)
* Update/Insert (for update with fallback to insert)
* Delete (for delete)

Control/objectid

SAP Business One object identifier

Control/keyname

Optional tags with the name of the key(s) for the SAP Business One object

### 1.3.3 B1 Service Schema

Example for approval request service

**<B1out type="service">**

<ApprovalRequest>

<Code>5</Code>

<ObjectType>112</ObjectType>

<IsDraft>Y</IsDraft>

<ObjectEntry>4</ObjectEntry>

<Status>arsApproved</Status>

<Remarks>Sell to SAP Labs China for TechEd 2009</Remarks>

<CurrentStage>1</CurrentStage>

<OriginatorID>4</OriginatorID>

<ApprovalRequestLines>

<ApprovalRequestLine>

<StageCode>1</StageCode>

<UserID>1</UserID>

<Status>ardApproved</Status>

<Remarks>OK11</Remarks>

</ApprovalRequestLine>

</ApprovalRequestLines>

<ApprovalRequestDecisions>

<ApprovalRequestDecision>

<ApproverUserName>manager</ApproverUserName>

<ApproverPassword>1234</ApproverPassword>

<Status>ardApproved</Status>

<Remarks>approved by vPlatform</Remarks>

</ApprovalRequestDecision>

</ApprovalRequestDecisions>

</ApprovalRequest>

**</B1out>**

**Prerequisites**

You have defined the service identifier, service method type, service method identifier; get method identifier, request structure and request key in the outbound setup of the scenario step design, selecting *Scenarios*→*Step Design* → *[Outbound]*.

B1out/@type

Mandatory B1out root tag, mandatory type attribute with the value service

### 1.3.4 B1 Service (Full) Schema

**<B1out type="service\_full">**

**<Control>**

<serviceid>ApprovalRequestsService</serviceid>

<mtype>update</mtype>

<method>UpdateRequest</method>

<getmethod>GetRequest</getmethod>

<requeststr>ApprovalRequestParams</requeststr>

<keyname>Code</keyname>

**</Control>**

**<Payload> <!—for add or update -->**

<ApprovalRequest>

<Code>5</Code>

<ObjectType>112</ObjectType>

<IsDraft>Y</IsDraft>

<ObjectEntry>4</ObjectEntry>

<Status>arsApproved</Status>

<Remarks>Sell to SAP Labs China for TechEd 2009</Remarks>

<CurrentStage>1</CurrentStage>

<OriginatorID>4</OriginatorID>

<ApprovalRequestLines>

<ApprovalRequestLine>

<StageCode>1</StageCode>

<UserID>1</UserID>

<Status>ardApproved</Status>

<Remarks>OK11</Remarks>

</ApprovalRequestLine>

</ApprovalRequestLines>

<ApprovalRequestDecisions>

<ApprovalRequestDecision>

<ApproverUserName>m</ApproverUserName>

<ApproverPassword>1234</ApproverPassword>

<Status>ardApproved</Status>

<Remarks>approved by vPlatform</Remarks>

</ApprovalRequestDecision>

</ApprovalRequestDecisions>

</ApprovalRequest>

**</Payload>**

**</B1out>**

B1out/@type

Mandatory B1out root tag, mandatory type attribute with the value service\_full

serviceid

Identifier/name of the SAP Business One service

mtype

An SAP Business One service provides multiple methods. The relevant methods you typically use in the outbound phase are methods to insert, update or delete a message. Define, which of the following actions you want to process:

* add
* update
* remove

method

With the tag you specify the identifier/name of the method, you want to process.

getmethod

Before performing the required action, the DI API of SAP Business One first retrieves data using the Get method. The service provides it. This is, for example, necessary, if you want to merge incoming data with already existing data in the B1 database for an update. Specify here the identifier/name of the Get method.

requeststr

The B1 service provides a special structure to hand over parameters, typically key information. Define the identifier/name of the structure.

keyname

Specify the identifier of the primary key for the service.

### 1.3.5 B1i SQL Schema

**<B1out type="b1isql">**

**<SQL sqlmode="single/multiple">**

<Table id="name" keylist="f1,f2,..." task="A" del="">

<Field id="name" value="field\_value" wrapchar="true"/>

<Field id="fieldname" value="field\_value" wrapchar=" false"/>

...

<Table id="name" keylist="f1,f2,..." task="I" del="">

<Field id="name" value="field\_value" wrapchar="true "/>

<Field id="name" value="field\_value" wrapchar="true "/>

...

</Table>

...

</Table>

<Table id="name" keylist="f1,f2,..." task="D" del="">

...

</Table>

...

**</SQL>**

**</B1out>**

B1out/@type

Mandatory B1out root tag, mandatory attribute with the value b1isql

SQL/@sqlmode

This is the mandatory tag SQL. The sqlmode attribute is also mandatory. Set the value to single or multiple. The flag is relevant for processing multiple SQL statements. The default is multiple.

* single

The integration framework concatenates multiple SQL statements with a semicolon. The system hands over the single SQL statement to the database. Most databases support the feature.

* multiple

In this mode B1i processes each SQL statement one after the other. This is necessary if the destination database system does not support semicolon-separated multiple SQL statements. This is, for example, true for the DB2 database system.

Table

Table is a fixed value. Use it to define a database table record. You can define multiple tags. You can nest the tags to represent a father-child relationship.

Table/@id

The id attribute specifies the name of the database table.

Table/@keylist

The keylist attribute represents the specification of the primary key field name. If the primary key consists of multiple concatenated field values, enter a comma-separated list of all field names.

Table/@task

Specify the operation in the database.

task="A" (Automatic detection)

Automatic detection decides whether an update or insert is necessary. The integration framework detects the correct operation for each record separately, using the definitions of the primary keys.

task="I" (Insert)

The operation is insert. If the record already exists, this leads to duplicate entries or a unique key constraint in the database.

task="U" (Update)

The operation is update. If the record does not exist, the integration framework does not insert any record.

task="D" (Delete)

The operation is delete. If the record does not exist, nothing happens.

Table/@del

If the operation is "D", define a comma-separated list of table names where the additional deletion happens. The prerequisite for cascading deletion is the same primary key.

Field

“Field” is a fixed value for defining a field in the database table record. You can define multiple of these tags.

Field/@id

The “id” attribute specifies the name of the database table field.

Field/@value

The “value” attribute contains the value of the field.

Field/@wrapchar

It is the default that the value is wrapped by the apostrophe character. If you do not want this because, for example, if the value contains a number, set wrapchar to *false*.

### 1.3.6 B1 SQL Schema

**<B1out type="sql">**

<sql>sql statement</sql>

<sql>sql statement</sql>

<sql>sql statement</sql>

...

**</B1out>**

**Prerequisites**

You have defined the SQL mode is defined in the outbound setup of the scenario step design, selecting *Scenarios* → *Step Design* → *[Outbound]* → *[Details]*.

B1out/@type

Mandatory B1out root tag, mandatory attribute with the value sql

sql

With the tag you introduce an SQL statement. You can define multiple of these tags. The integration framework generates the SQL statements for the *B1isql* format, refer to the previous chapter. Using SQL, the vBIU finally generates the SQL statement. The integration framework sends the statement without any interpretation to the database system.

### 1.3.7 SQL (Full) Schema

**<B1out type="sql">**

<Control>

<sqlmode>single</sqlmode>

</Control>

<sql>sql statement</sql>

<sql>sql statement</sql>

<sql>sql statement</sql>

...

**</B1out>**

B1out/@type

Mandatory B1out root tag, mandatory attribute with value sql\_full

sqlmode

Mandatory tag SQL. The sqlmode attribute is also mandatory. The flag is relevant for processing multiple SQL statements. The default is multiple.

single

The integration framework concatenates multiple SQL statements with a semicolon. The integration framework hands over the single SQL statement to the database. Most databases systems support the feature.

multiple

In this mode the integration framework processes each SQL statement one after the other. This is necessary if the destination database system does not support semicolon-separated multiple SQL statements. This is, for example, true for the DB2 database system.

sql

With the tag you introduce an SQL statement. You can define multiple of these tags. The integration framework generates the SQL statements for the *B1isql* format, refer to the previous chapter. Using SQL, the vBIU finally generates the SQL statement. The integration framework sends the statement without any interpretation to the database system.

## 1.4 Database

### 1.4.1 B1i SQL Database Schema

**<DBout type="b1isql">**

**<SQL sqlmode="single/multiple">**

<Table id="name" keylist="f1,f2,..." task="A" del="">

<Field id="name" value="field\_value" wrapchar="true"/>

<Field id="fieldname" value="field\_value" wrapchar=" false"/>

...

<Table id="name" keylist="f1,f2,..." task="I" del="">

<Field id="name" value="field\_value" wrapchar="true "/>

<Field id="name" value="field\_value" wrapchar="true "/>

...

</Table>

...

</Table>

<Table id="name" keylist="f1,f2,..." task="D" del="">

...

</Table>

...

**</SQL>**

**</DBout>**

DBout/@type

Mandatory DBout root tag, mandatory attribute with the value b1isql

SQL/@sqlmode

Mandatory SQL tag. The sqlmode attribute is also mandatory. This flag is relevant for processing multiple SQL statements. The default is multiple.

single

The integration framework concatenates multiple SQL statements with a semicolon. The integration framework hands over the single SQL statement to the database. Most databases systems support the feature.

multiple

In this mode the integration framework processes each SQL statement one after the other. This is necessary if the destination database system does not support semicolon-separated multiple SQL statements. This is, for example, true for the DB2 database system.

Table

It is a fixed value to define a database table record. You can define multiple tags. You can nest tags to represent a father-child relationship.

Table/@id

The *id* attribute specifies the name of the database table.

Table/@keylist

The *keylist* attribute represents the specification of the primary key field name. If the primary key consists of multiple concatenated field values, specify a comma-separated list of all field names.

Table/@task

Specify the operation in the database.

task="A" (Automatic detection)

Automatic detection decides whether an update or insert is necessary. The integration framework detects the correct operation for each record separately using the definitions of the primary keys.

task="I" (Insert)

The operation is insert. If the record already exists, this leads to duplicate entries or a unique key constraint in the database.

task="U" (Update)

The operation is update. If the record does not exist, the integration framework does not insert any record.

task="D" (Delete)

The operation is delete. If the record does not exist, nothing happens.

Table/@del

If the operation is D, define a comma-separated list of table names where the additional deletion happens. The prerequisite for this cascading deletion is the same primary key.

Field

This is a fixed value for defining a field in the database table record. You can define multiple of these tags.

Field/@id

The *id* attribute specifies the name of the database table field.

Field/@value

The *value* attribute contains the value of the field.

Field/@wrapchar

It is the default that the value is wrapped by the apostrophe character. If you want to avoid this, if, for example, the value contains a number, set wrapchar to false.

### 1.4.2 SQL Database Schema

<DBout type="sql">

<sql>sql statement</sql>

<sql>sql statement</sql>

<sql>sql statement</sql>

...

</DBout>

**Prerequisites**

SQL Mode is defined in the outbound setup of the Scenario Step design (*Scenarios*→*Step Design* → *[Outbound]*).

DBout/@type

Mandatory DBout root tag, mandatory attribute with the value sql

sql

With this tag you introduce an SQL statement. You can define multiple of these tags. The integration framework generates the SQL statements for the *B1isql* format, refer to the previous chapter. Using SQL, the vBIU finally generates the SQL statement. The integration framework sends the statement without any interpretation to the database system.

### 1.4.3 SQL (Full) Database Schema

<DBout type="sql\_full">

<Control>

<sqlmode>single</sqlmode>

</Control>

<sql>sql statement</sql>

<sql>sql statement</sql>

<sql>sql statement</sql>

...

</DBout>

DBout/@type

Mandatory DBout root tag, mandatory attribute with the value sql\_full

sqlmode

Mandatory tag SQL. The sqlmode attribute is also mandatory. This flag is relevant for processing multiple SQL statements. The default is multiple.

single

The integration framework concatenates multiple SQL statements with a semicolon. The integration framework hands over this single SQL statement to the database. Most databases systems support this feature.

multiple

In this mode the integration framework processes each SQL statement one after the other. This is necessary if the destination database system does not support semicolon-separated multiple SQL statements. This is, for example, true for the DB2 database system.

sql

With this tag you introduce an SQL statement. You can define multiple of these tags. The integration framework generates the SQL statements for the *B1isql* format, refer to the previous chapter. Using SQL, the vBIU finally generates the SQL statement. The integration framework sends the statement without any interpretation to the database system.

## 1.5. Flat File

### 1.5.1 XML Flat File Schema

You can use any well-formed XML payload.

**Prerequisites**

You have defined the following configuration in the outbound setup of the scenario step design, selecting *Scenarios*→*Step Design* →[Outbound] →[Details]:

* Outbound Format: xml

### 1.5.2 DSV Flat File Schema

<Fileout type="file">

<row>

<col>value</col>

<col>value</col>

...

</row>

...

<row>

<col>value</col>

<col>value</col>

...

</row>

</Fileout>

**Prerequisites**

You have defined the following configuration in the outbound setup of the scenario step design, selecting *Scenarios*→*Step Design* →[Outbound] →[Details]:

* Outbound Format: dsv (delimiter-separapted values)
* DSV Field Delimiter: ,

Optional Settings:

You have defined the following configuration in the outbound setup of the scenario step design, selecting *Scenarios*→*Step Design* →[Outbound] →[Details]:

* Character Encoding: character encoding (default is ISO-8859-1)
* DSV Field Wrapper: Add a wrapper character to wrap a field in case a value contains a delimiter.

row

This tag indicates a row. You can use multiple of these tags.

col

This tag indicates a column. You can have multiple of these tags. The integration framework concatenates the values of these tags with the delimiter character in one line, following the order of the col elements.

### 1.5.3 DSV (Full) Flat File Schema

<Fileout type="file\_full">

<Control>

<fmode>write</fmode>

<format>dsv</format>

<encoding></encoding>

<deli>,</deli>

<wrap>"</wrap>

</Control>

<row>

<col>value</col>

<col>value</col>

...

</row>

...

<row>

<col>value</col>

<col>value</col>

...

</row>

</Fileout>

fmode

This tag defines the write mode for the outbound adapter.

* append creates the file if it does not exist and appends the file, if it already exists.
* write creates the file if it does not exist and throws an exception if the file already exists.
* overwrite writes the file, no matter if it already exists or not.
* delete deletes the file whether it already exists or not. The attempt to delete a file that does not exist is no error. It is however an error, if the deletion of an already existing file fails.
* rename

format

Specify the outbound format. Enter dsv.

encoding

Select the required character encoding from the list. The default is ISO-8859-1.

Character encoding is necessary to apply the technical representation of the characters according to the country or system-specific needs. The following entries are available: ISO 8859-1, ISO 8859-2, ISO 8859-3, ISO 8859-4, ISO 8859-5, ISO 8859-6, ISO 8859-7, ISO 8859-8, ISO 8859-9, ISO 8859-11, ISO 8859-13, ISO 8859-14, ISO 8859-15, ISO 8859-16, US-ASCII, EBCDIC, Shift-JIS, EUC-JP, ISO-2022, GB2312, EUC-KR, Big5, Unicode UTF-7, Unicode UTF-8, Unicode UTF-16, ISO-10646-UCS2, ISO-10646-UCS4.

deli

Enter the delimiter for the field list. The default delimiter for DSV files is a comma.

wrap

Specify the field wrapper character. The field wrapper is optional. Use it only for values containing the delimiter character, for example, the comma.

row

This indicates a row. You can use multiple of these tags.

col

This indicates a column. You can have multiple of these tags. The integration framework concatenates the values of these tags with the delimiter character in one line, following the order of the col elements.

### 1.5.4 Text Flat File Schema

You can use any well-formed XML payload.

Define the following format control document:

<FileOutbound>

<segment id="name">

<field id="" from="" len=" " default="" justify="" fillchar=""/>

<field id="" from="" len="" default="" justify="" fillchar=""/>

...

<segment id="name">

<field id="" from="" len="" default="" justify="" fillchar=""/>

<field id="" from="" len="" default="" justify="" fillchar=""/>

...

<segment id="name">

<field id="" from="" len="" default="" justify="" fillchar=""/>

<field id="" from="" len="" default="" justify="" fillchar=""/>

...

</segment>

</segment>

</segment>

...

</FileOutbound>

**Prerequisites**

You have defined the following configuration in the outbound setup of the scenario step design, selecting *Scenarios*→*Step Design* →[Outbound] →[Details].

* Outbound Format: txt
* Format Control Document: Name of the document containing the rules in the scenario step group in the BizStore.

**Optional Settings**

You have defined the following configuration in the outbound setup of the scenario step design, selecting *Scenarios*→*Step Design* →[Outbound] →[Details].

* Character Encoding: character encoding (default is ISO-8859-1)

| **Element** | **Description** |
| --- | --- |
| FileOutbound | Mandatory root tag |
| segment | Text outbound supports the definition of multiple line structures. Each segment describes the structure of one line. |
| segment/@id | Identifier of a segment. The integration framework processes all elements in the outbound XML document with this name following the specified rules. The integration framework generates the outbound file based on the order of the segment definitions. |
| field | In a segment, you can define multiple fields and also segments. |
| field/@id | Field identifier. For processing all sub elements with this name inside the element corresponding to the segment are relevant in the outbound XML. |
| field/@from | Defines the column where the field starts in the outbound text document |
| field/@len | Defines the field length in the outbound text document. |
| field/@default | Define a default value. If such a value exists, it overwrites the value provided by the scenario step |
| field/@justify | Define whether the integration framework fills the field from the left side (the field is right aligned (R) or from the right side (the field is left aligned (L). |
| field/@fillchar | Define the character, the integration framework fills into the field. If the definition for fillchar is empty, the integration framework fills the field with blank. |

For the default field, you can use the following variables:

|  |  |
| --- | --- |
| **Name** | **Description** |
| [lcnt]: line counter | Counts the number for each line |
| [mcnt]: main counter | If a segment is a sub-segment, the integration framework places the line counter of the father segment here. |
| [yyyy] | The integration framework replaces the variable with the current year (four digits). |
| [mm] | The integration framework replaces the variable with the current month (two digits) |
| [dd] | The integration framework replaces the variable with the current day (two digits). |
| [hour] | The integration framework replaces the variable with the current hour (two digits) |
| [min] | The integration framework replaces the variable with the current minute (two digits) |
| [sec] | The integration framework replaces the variable with the current second (two digits) |

### 1.5.5 Text (Full) Flat File Schema

<Fileout type="file\_full">

<Control>

<fmode>write</fmode>

<format>txt</format>

<encoding></encoding>

<ruledoc>myrule.xml</ruledoc>

</Control>

Any wellformed xml payload

</Fileout>

Provide additionally the following rule document:

<FileOutbound>

<segment id="name">

<field id="" from="" len=" " default="" justify="" fillchar=""/>

<field id="" from="" len="" default="" justify="" fillchar=""/>

...

<segment id="name">

<field id="" from="" len="" default="" justify="" fillchar=""/>

<field id="" from="" len="" default="" justify="" fillchar=""/>

...

<segment id="name">

<field id="" from="" len="" default="" justify="" fillchar=""/>

<field id="" from="" len="" default="" justify="" fillchar=""/>

...

</segment>

</segment>

</segment>

...

</FileOutbound>

| **Tag** | **Description** |
| --- | --- |
| fmode | Define the write mode for the outbound adapter.   * append creates the file if it does not exist and appends the file, if it already exists. * write creates the file if it does not exist and throws an exception if the file already exists. * overwrite writes the file, no matter if it already exists or not. * delete deletes the file whether it already exists or not. The attempt to delete a file that does not exist is no error. It is however an error, if the deletion of an already existing file fails. * rename |
| format | Define the outbound format. Enter txt. |
| encoding | Define the encoding. The default is ISO-8859-1.  Character encoding is necessary to apply the technical representation of the characters according to the country or system specific needs. The following entries are available: ISO 8859-1, ISO 8859-2, ISO 8859-3, ISO 8859-4, ISO 8859-5, ISO 8859-6, ISO 8859-7, ISO 8859-8, ISO 8859-9, ISO 8859-11, ISO 8859-13, ISO 8859-14, ISO 8859-15, ISO 8859-16, US-ASCII, EBCDIC, Shift-JIS, EUC-JP, ISO-2022, GB2312, EUC-KR, Big5, Unicode UTF-7, Unicode UTF-8, Unicode UTF-16, ISO-10646-UCS2, ISO-10646-UCS4. |
| ruledoc | Define the name of the format control document: This document must be available in the scenario step group in the BizStore. |

## 1.6 Web Service

queryparam="query"

Any well-formed XML payload

**Optional**

You have defined the following configuration in the outbound setup of the scenario step design, selecting *Scenarios*→*Step Design* →[Outbound] →[Details].

* SOAP Action: soap action definition

queryparam

Optionally define a URL query parameter. You can directly define it as an attribute in the first line of the vBIU outbound section.

<xsl:template name="transform">

<xsl:attribute name="queryparam">querykey.K0=12, querykey.K1=A</xsl:attribute>

The format is a comma-separated list of key-value pairs in the format querykey.<keyname>=<value>. The example above leads to a call of the following destination url: url?K0=12&K1=A

SOAP Action

It is the default that the integration framework hands over an empty SOAP action. This serves most of the cases. There are however some older Web services that need the definition of the called method in the SOAP action. For this purpose you can define the SOAP action in the outbound definition.

# 2. Process Atoms

## 2.1 Atom Send E-Mail Schema

<email>

<connect>

<host>mailsmtpserver</host>

<port>25</port>

<user>user</user>

<password>password</password>

</connect>

<envelope>

<to>

<address>a1.b@x.com</address>

...

</to>

<cc>

<address>c1.b@x.com</address>

...

</cc>

<bcc>

<address>b1.b@x.com</address>

...

</bcc>

<from>ab.sender@sap.com</from>

<subject>my subject</subject>

<message>This is a test</message>

<attachment doc="/ds/grp/doc" pltype="jpg"/>

...

</envelope>

</email>

## 2.2 Atom HTTP Call Schema

<call xmlns="">

<query id="identifier" value="value"/>

...

<connect>

<destProtocol>https</destProtocol>

<destHost>streamwork.com</destHost>

<destPort>443</destPort>

<destPath>oauth/request\_token</destPath>

<query/>

<proxyHost>proxy</proxyHost>

<proxyPort>8080</proxyPort>

<method>POST</method>

<authentification>none</authentification>

<user/>

<password/>

<user2query/>

<password2query/>

<sslTruststorePath/>

<sslTruststorePassword/>

</connect>

<pltype> </pltype>

<payload>

...

</payload>

<htta>

<par id="httpheader.???" value="value"/>

...

<par id="htta.returnpltypeforce" value=""/>

<par id="htta.returnpltypedefault" value=""/>

</htta>

</call>

htta.returnpltypedefault

This redefines the default payload-type of the returning data. This default only applies if a payload-type is not enforced or cannot be determined by the MIME-type of the received data. It prevents the adapter to fall back to the otherwise inbuilt default, which is to flag the data as general binary content. If htta.reporterror is set to true and an error occurs in the HTTA-call, the payload-type is xml.

htta.returnpltypeforce

This enforces the payload-type of the returning data. This setting supersedes the payload-type determined by the MIME-type of the received data. If htta.reporterror is set to true and an error occurs in the HTTA-call, the payload-type is xml. Only the definition of the aux-in property bpm.pltype can supersede this definition.

Avoid using bpm.pltype, if the expected regular payload-type is not XML and htta.reporterror is set to true: In such a case you need a situation-conditional payload-type determination. This is not possible using bpm.pltype!

httpheader.???

All properties that have the httpheader prefix, become HTTP header fields in the message. The integration framework cuts out the prefix of the resulting header name.

## 2.3 Atom .NET Call Schema

<dotnetcall xmlns="">

<Assembly title="String" path="String" />

<Function name="String" />

<ParameterList>

<Parameter dataType="String" isArray="Boolean">String</Parameter>

<!-- 0 or many of these tags are allowed -->

<!-- enumeration for dataType: sbyte,byte,short,ushort,int,uint,

long,ulong,char,float,double,bool,decimal,string -->

</ParameterList>

<RedirectConsoleOutput>true</RedirectConsoleOutput>

</dotnetcall>

Assembly Tag, title Attribute

Enter the name of the assembly you want to call. Provide the title with namespace and class name in the following way: <namespace>.<classname>

Assembly Tag, path Attribute

Enter the absolute or relative path to the assembly you want to call.

Function Tag, name Attribute

Enter the name of the function you want to call.

ParameterList Tag, Parameter Tag, dataType Attribute

Enter the datatype of the parameter or parameters.

Use only primitive data types like sbyte, byte, short, ushort, int, uint, long, ulong, char, float, double, bool, decimal, string

ParameterList Tag, Parameter Tag, isArray Attribute

If you want to enter one value, set the isArray value to false. If you want to enter an array of values, set the isArray value to true.

ParameterList Tag, Parameter Tag, Value Attribute

If isArray is set to false, use one <Value> tag to enter the value of the parameter.

If the parameter is an array (isArray is set to true), enter the <Value> tags.

Enter all mandatory parameter values of the function you want to call. Enter the values in the sequence the function requires.

RedirectConsoleOutput

If you want to add the .NET console output to the outbound message, set parameter RedirectConsoleOutput to true. Otherwise set it to false.

## 2.4 Correct after Branch

<Msg xmlns="urn:com.sap.b1i.vplatform:entity">

<xsl:copy-of select="/bfa:unbranch/vpf:Msg[1]/@\*"></xsl:copy-of>

<xsl:copy-of select="/bfa:unbranch/vpf:Msg[1]/vpf:Header"></xsl:copy-of>

<Body>

<xsl:copy-of select="/bfa:unbranch/vpf:Msg[1]/vpf:Body/\*"></xsl:copy-of>

<Payload Role="R" id="{$atom}" ts="{$vpts}">

<xsl:call-template name="transform"></xsl:call-template>

</Payload>

</Body>

</Msg>

The unbranch element consolidates the results of all paths that the integration framework processes. To obtain a well-formed XML document, the unbranch element adds an artificial <bfa:unbranch> root tag to the integration framework message to consolidate all results from the different path sequences. Consider this in the first subsequent transformation atom after the unbranch element. To correct unbranch, click [Generate] in the transformation atom and select the Correct after Branch option.

## 2.5 Payload Type JSON

<xsl:template name="B1if.pltype\_json">

<!--option 1 - starting in the root with an object-->

<io xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   
 xmlns="urn:com.sap.b1i.bizprocessor:bizatoms" pltype="json"   
 xsi:schemaLocation="urn:com.sap.b1i.bizprocessor:bizatoms json\_pltype.xsd">

<object>

<!--optional multiple of the following elements-->

<string name="">value</string>

<number name="">value</number>

<object name="">...</object>

<array>...</array>

<bool name="">true/false</bool>

<null name=""></null>

</object>

</io>

<!--option 2 - starting in the root with an array-->

<io xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   
 xmlns="urn:com.sap.b1i.bizprocessor:bizatoms" pltype="jso"   
 xsi:schemaLocation="urn:com.sap.b1i.bizprocessor:bizatoms json\_pltype.xsd">

<array>

<!--optional multiple of the following elements-->

<string name="">value</string>

<number name="">value</number>

<object name="">...</object>

<array>...</array>

<bool name="">true/false</bool>

<null name=""></null>

</array>

</io>

</xsl:template>

# 3. Preconfiguration Schemas

## 3.1 Scenario Package – Setup Description Schema

If you use the integration framework API to automatically activate a scenario package, you need the following XML document.

For more information, refer to guide 05 about APIs, section 4.

<vPacSetup xmlns="urn:com.sap.b1i.vplatform:entity" Id="package id" Status="status">

<SenderList>

<Sender id="sysid" active="true"/>

<Sender id="sysid" active="true"/>

...

</SenderList>

<ReceiverList>

<Receiver id="sysid" active="true"/>

<Receiver id="sysid" active="true"/>

...

</ReceiverList>

<vBIUList>

<vBIU Id="step identifier" active="true"/>

<vBIU Id="step identifier" active="true"/>

...

</vBIUList>

<Publish>

<vBIU Id="step identifier">

<Sender Id="sysid" filter="" active="true"/>

<Sender Id="sysid" filter="" active="true"/>

...

</vBIU>

<vBIU Id="step identifier">

<Sender Id="sysid" filter="" active="true"/>

<Sender Id="sysid" filter="" active="true"/>

...

</vBIU>

...

</Publish>

<Subscriptions>

<vBIU Id="step identifier">

<Receiver Id="sysid" filter="" active="true"/>

<Receiver Id="sysid" filter="" active="false"/>

...

</vBIU>

<vBIU Id="step identifier">

<Receiver Id="sysid" filter="" active="true"/>

<Receiver Id="sysid" filter="" active="false"/>

...

</vBIU>

...

</Subscriptions>

<TimerList>

<Timer Id="step identifier" min="" hour="" day="" month="" dow="" year=""/>

<Timer Id="step identifier" min="" hour="" day="" month="" dow="" year=""/>

...

</TimerList>

</vPacSetup>

SenderList

List all sender systems, relevant for all scenario steps you want to set up. You can define unique identifier (Sysids) for the systems in the System Landscape Directory (SLD).

For more information, refer to guide 05 about APIs, section 2.

ReceiverList

List all receiver systems, relevant for all scenario steps you want to set up. You can define unique identifier (Sysids) for the systems in the SLD.

For more information, refer to guide 05 about APIs, section 2.

Publish

Specify all publications. For each scenario step, you want to set up, create a section <vBIU>. In this section list all sender systems you want to set up for this particular scenario step. Use the attribute filter to specify a condition under which the integration framework processes the sender message. You can use any xPath statement. If the inbound is *Predecessor*, specify the Sysid of the predecessor scenario step. If the inbound is *Internal Queue*, specify the internal integration framework SysId 0010000000.

Subscriptions

Specify all subscriptions. For each scenario step you set up to publish data, and which is an asynchronous step, create a section <vBIU>. Inside this section list all receiver systems you want to send data to. With the attribute filter specify a condition under which the receiver receives the message. You can use any xPath statement. If the outbound is *Void*, specify the internal integration framework SysId 0010000000.

TimerList

Specify scheduler settings for all scenario steps triggered by a timer. This definition is optional. If there is no definition, the integration framework takes the timer settings of the scenario step design.

# Copyrights, Trademarks, and Disclaimers

© Copyright 2018 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.