



Integration Guide | PUBLIC

Document Version: 1.1 – 2025-01-17

# **SAP IBP - Reusable Integration Flows**

**Data Integration Between SAP IBP for Supply Chain 2411 and Higher and SAP Cloud Integration Using SAP IBP - Reusable Integration Flows**

# Content

- 1 Document History. . . . . 3**
- 2 Introduction. . . . . 4**
- 3 Prerequisites. . . . . 5**
- 4 SAP IBP Read Reusable Integration Flows. . . . . 6**
  - 4.1 SAP IBP Read - Initialize. . . . . 6
    - Configuring the SAP IBP Read - Initialize Reusable Integration Flow. . . . . 7
    - Message Body Attributes for the SAP IBP Read - Initialize Reusable Integration Flow. . . . . 10
    - Reading Various Data Types. . . . . 11
    - Attachments and Error Logs of the SAP IBP Read - Initialize Reusable Integration Flow. . . . . 13
  - 4.2 SAP IBP Read - Fetch Data. . . . . 13
    - Configuring the SAP IBP Read - Fetch Data Reusable Integration Flow. . . . . 14
    - Message Body Attributes for the SAP IBP Read - Fetch Data Reusable Integration Flow. . . . . 16
    - Attachments and Error Logs of the SAP IBP Read - Fetch Data Reusable Integration Flow. . . . . 17
  - 4.3 SAP IBP Read - Close and SAP IBP Read - Cancel. . . . . 18
    - Configuring the SAP IBP Read - Close and the SAP IBP Read - Cancel Reusable Integration Flow. . . . . 18
    - Attachments and Error Logs of the SAP IBP Read - Close and the SAP IBP Read - Cancel Reusable Integration Flow. . . . . 21
  - 4.4 Recommendations. . . . . 21
- 5 SAP IBP Write Reusable Integration Flows. . . . . 23**
  - 5.1 SAP IBP Write - Create Batches. . . . . 24
    - Configuring the SAP IBP Write - Create Batches Reusable Integration Flow. . . . . 24
    - Message Body Attributes for the SAP IBP Write - Create Batches Reusable Integration Flow . . . . . 27
    - Writing Various Data Types. . . . . 28
    - Error Handling. . . . . 28
  - 5.2 SAP IBP Write - Post Data. . . . . 29
    - Configuring the SAP IBP Write - Post Data Reusable Integration Flow. . . . . 31
    - Message Body Attributes for the SAP IBP Write - Post Data Reusable Integration Flow. . . . . 34
  - 5.3 SAP IBP Write - Process Posted Data. . . . . 35
    - Configuring SAP Write - Process Posted Data Reusable Integration Flow. . . . . 35
    - Error Handling. . . . . 36
  - 5.4 Recommendations. . . . . 37

# 1 Document History

The following table provides an overview of the most important changes.

Version	Date	Description
1.1	January 17, 2025	Extended the <a href="#">Configuring the SAP IBP Write - Post Data Reusable Integration Flow [page 31]</a> and the <a href="#">Message Body Attributes for the SAP IBP Write - Post Data Reusable Integration Flow [page 34]</a> sections with the <code>DataFormat</code> attribute.
1.0	November 25, 2024	Initial version

## 2 Introduction

You can transfer data between SAP Integrated Business Planning for Supply Chain (SAP IBP) and SAP Cloud Integration using *SAP IBP - Reusable Integration Flows*.

Data integration between SAP IBP and SAP Cloud Integration using the artifacts in the *SAP IBP - Reusable Integration Flows* package is available in 2411 and higher.

To read data from SAP IBP, you can use the following *SAP IBP - Reusable Integration Flows*:

- *SAP IBP Read - Initialize*
- *SAP IBP Read - Fetch Data*
- *SAP IBP Read - Close*
- *SAP IBP Read - Cancel*

Use the following *SAP IBP - Reusable Integration Flows* to write data in SAP IBP:

- *SAP IBP Write - Create Batches*
- *SAP IBP Write - Post Data*
- *SAP IBP Write - Process Posted Data*

# 3 Prerequisites

You can transfer data if you have configured the connection between SAP Integrated Business Planning for Supply Chain (SAP IBP) and SAP Cloud Integration as follows:

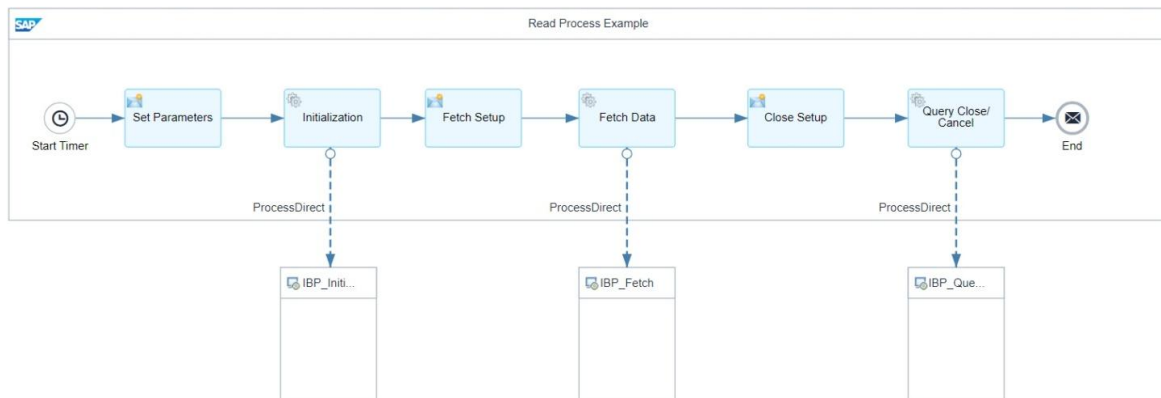
- You've enabled the *Planning - Integration Suite-Cloud Integration Integration* (SAP\_COM\_0931) communication scenario that allows a connection between SAP IBP and SAP Cloud Integration. Basic authentication and authentication with certificate are available for this communication scenario. For more information on the procedure, see [Setting Up the Integration for Using SAP IBP - Reusable Integration Flows](#).
- You've configured SAP BTP/SAP Integration Suite to connect to SAP IBP using the communication arrangement described in [Setting Up the Integration for Using SAP IBP - Reusable Integration Flows](#).

# 4 SAP IBP Read Reusable Integration Flows

Call the **SAP IBP Read** reusable integration flows in the correct sequence to transfer data from SAP IBP to SAP Cloud Integration.

1. Call *SAP IBP Read - Initialize* to initialize the query.  
The initialization triggers an asynchronous query.
  - During the initialization process, the system assembles a query based on the type of data, column selection, filtering and sorting options.
  - The system runs the query and stores the result in a new staging table.
  - SAP Cloud Integration monitors the status at regular intervals, and once the staging table is created, the process can proceed to the next step.
2. When calling the *SAP IBP Read - Fetch Data* reusable integration flow, SAP Cloud Integration retrieves data in packages which SAP IBP provides from a staging table.
3. Once data is fetched, *SAP IBP Read - Close* closes the query and deletes the staging table.  
You can cancel the query by calling *SAP IBP Read - Cancel*, for example, if an error occurs in the calling integration flow, outside the query. Canceling the query is equivalent to closing the operation.

## Result

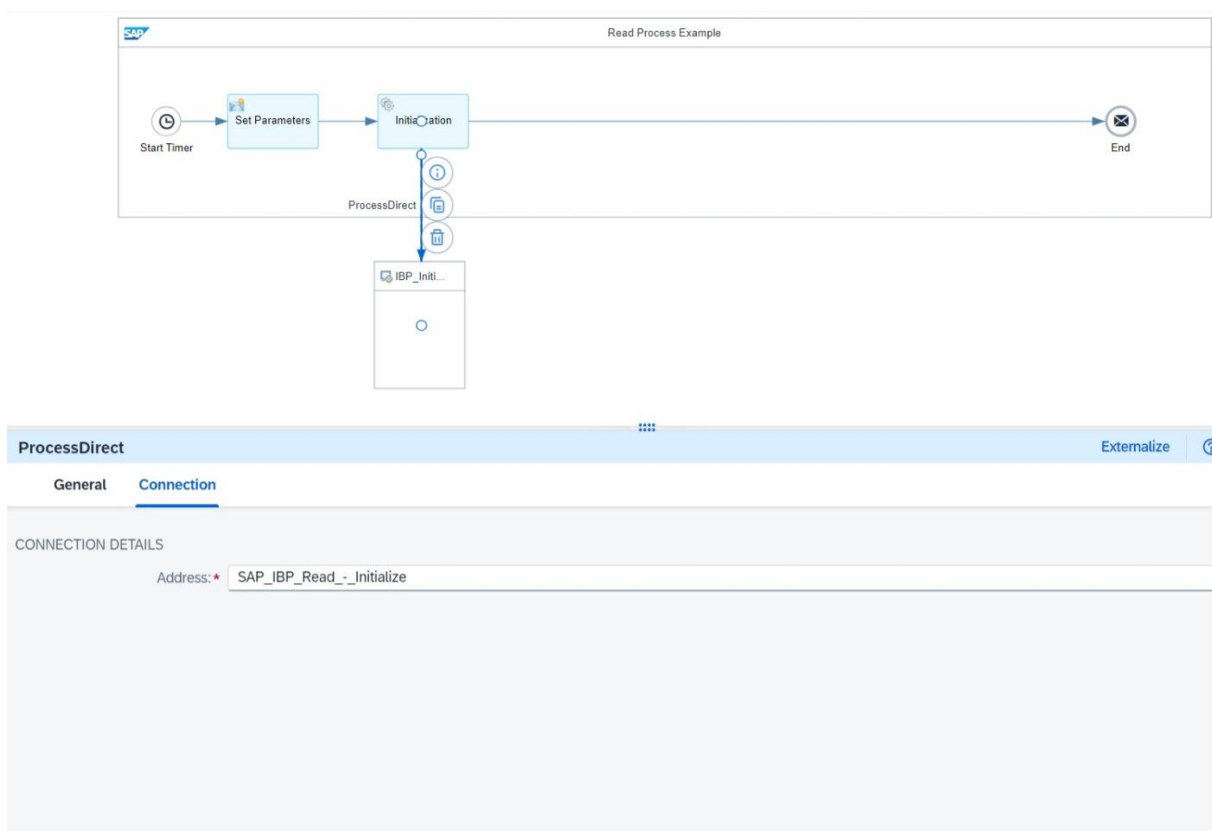


## 4.1 SAP IBP Read - Initialize

## 4.1.1 Configuring the SAP IBP Read - Initialize Reusable Integration Flow

### Context

To configure the *SAP IBP Read - Initialize* reusable integration flow, perform the following procedure. You can call this integration flow with the `SAP_IBP_Read_-_Initialize` process ID.



### Procedure

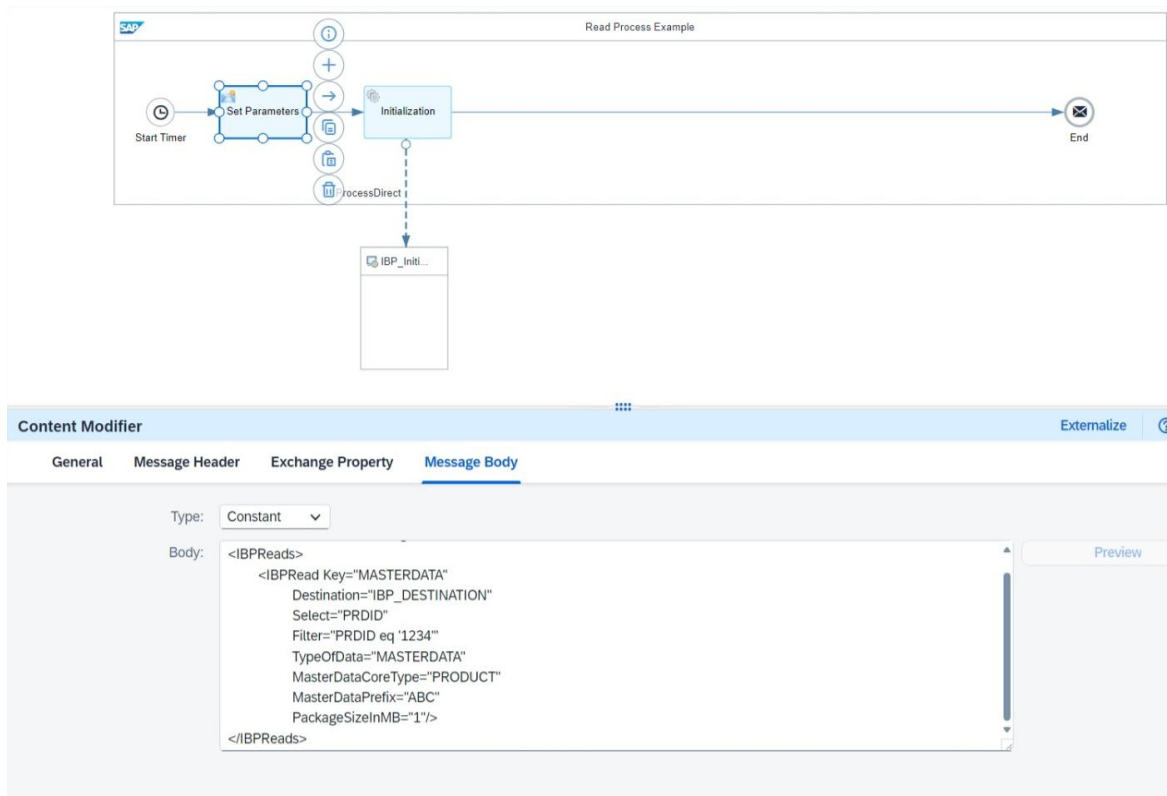
1. Initialize the query to read data from SAP Cloud Integration. Configure the input of the integration flow in the body of a Content Modifier in XML format.

Provide an `IBPRead` node. The properties of the SAP IBP Read node are used to pass the parameters.

This example shows how to read the `Product` master data type with 1234 product ID by filling the parameters in the Set Parameters step (Content Modifier).

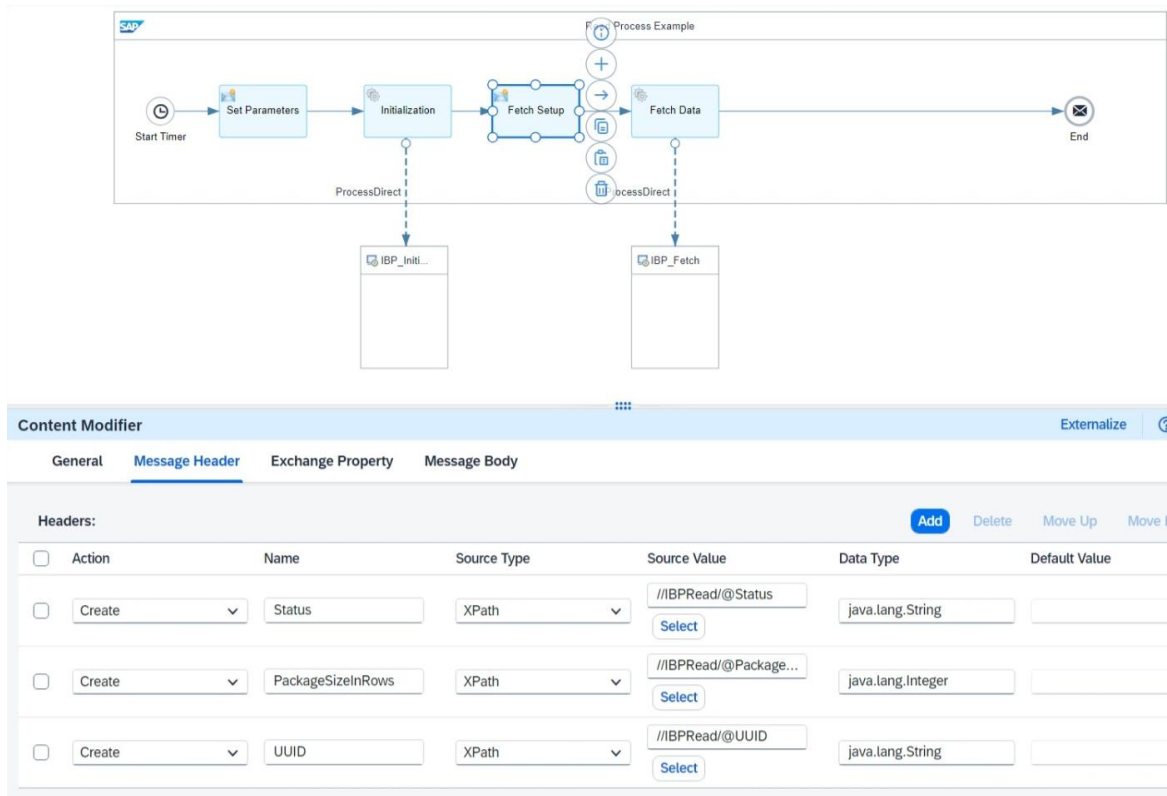
## Sample Code

```
Procedure
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<IBPReads>
  <IBPRead Key="MASTERDATA"
    Destination="IBP_DESTINATION"
    Select="PRDID"
    Filter="PRDID eq '1234'"
    TypeOfData="MASTERDATA"
    MasterDataCoreType="PRODUCT"
    MasterDataPrefix="IBP" />
</IBPReads>
```



2. Use the integration flow ID to run the integration flow with a Process Direct Call between a Request Reply and a Receiver.





## Results

Once the integration flow is complete, it returns a structure with information such as UUID, rows etc. This structure is available in the body. The result of the initialization is provided in the header and as an attachment with the name `IBPReadRequestsResults`. If the initialization was successful, the status returns `FETCH_DATA`. If unsuccessful, it returns `ERROR`.

### Output Code

```
?xml version="1.0" encoding="utf-8"?>
<IBPRead xmlns:multimap="
http://sap.com/xi/XI/SplitAndMerge"
  Destination="IBP_DESTINATION"
  Filter="PRDID eq '1234'"
  Index="1"
  Key="MASTERDATA"
  Limit="2147483647"
  MasterDataCoreType="PRODUCT"
  MasterDataPrefix="IBP"
  MaximumThreads="4"
  OrderBy=""
  PackageSizeInRows="0"
  Rows="1"
  SQLStatement="SELECT PRDID FROM (SELECT PRDID FROM
SAPSOPG100.SOPMD_IBPPRODUCT WHERE PRDID = '1234' LIMIT 2147483647)"
  Select="PRDID"
  Status="FETCH_DATA"
  TypeOfData="MasterData"
```

```
UUID="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" />
```

## 4.1.2 Message Body Attributes for the SAP IBP Read - Initialize Reusable Integration Flow

You can use the following attributes for the `IBPRead` tag when you call the SAP IBP Read - Initialize reusable integration flow.

Parameter Name	Description
<code>Key</code>	Identifies the specific integration flow when multiple <i>SAP IBP Read</i> integration flows are defined. This parameter is mandatory.
<code>Destination</code>	Defines the destination of the SAP IBP system that was previously created in the SAP BTP subaccount. It's a mandatory parameter.
<code>CallerName</code>	Specifies the name of the caller of the integration flow. Serves as identification in SAP IBP application logs.
<code>Select</code>	Defines a list of fields that you want to select, separated by commas. It's a mandatory parameter. For example, <code>PRDID, PRDGROUP</code> etc.
<code>OrderBy</code>	Lists data in a specified order, separated by commas. You can create a descending order by adding 'desc' after the field name.
<code>Filter</code>	Defines the filter criteria in restricted OData format. For more information, see <a href="#">Reading Various Data Types [page 11]</a> .
<code>TypeOfData</code>	Defines the type of data that you want to read. Accepted values are as follows: <code>MASTERDATA</code> , <code>KEYFIGURE</code> , <code>TIMEPROFILE</code>
<code>MasterDataPrefix</code>	Defines a prefix for the master data. It's only relevant for reading master data.
<code>MasterDataCoreType</code>	Defines the core type of SAP IBP master data. It's only relevant for reading master data.
<code>PlanningArea</code>	Specifies the planning area that you use in SAP IBP. It's relevant for key figures and version-specific master data.

Parameter Name	Description
PlanningAreaVersion	Defines the version of the planning area that you use in SAP IBP. It's relevant for version-specific key figures and version-specific master data.
TimeAggregationLevel	Determines the time aggregation level for reading key figures. It's only relevant for reading key figures.
FilterUserId	Defines the ID of the business user to apply user-specific filters. It's only relevant for reading key figures. You can use this parameter along with <code>FilterId</code> .
FilterId	Specifies the ID of a predefined filter to read key figures. It's only relevant for reading key figures. You can use this parameter along with <code>FilterUserId</code> .
TimeProfile	Defines the ID of the time profile that you want to read. It's only relevant for reading time profile data.
Limit	Sets the upper limit for the number of selected rows. Exceeding records are ignored without error. Default value: 2147483647
DetailedTraceLog	Activates the detailed trace log in SAP IBP. It also triggers more messages in the response from SAP IBP.
PackageSizeInMB	Specifies the number of megabytes for the package size that you want to receive.

## 4.1.3 Reading Various Data Types

By configuring the Content Modifier, you can extract different types of data from SAP IBP.

### Master Data

Fill in the `Key`, `TypeOfData`, `MasterDataPrefix`, `MasterDataCoreType`, `Destination` and the `Select` fields to read master data.

#### Sample Code

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<IBPReads>
  <IBPRead Key="Product "
           TypeOfData="MasterData "
           MasterDataPrefix="IBP"
           MasterDataCoreType="PRODUCT" />
</IBPReads>
```

```
Destination="IBP_DESTINATION"  
Select="PRDID,PRDDESCR" />  
</IBPReads>
```

## Key Figures

To read key figures, fill in the Key, TypeOfData, PlanningArea, Destination, TimeAggregationLevel and the Select fields.

### Sample Code

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>  
<IBPReads>  
  <IBPRead Key="ActualsQty"  
    TypeOfData="KeyFigures"  
    PlanningArea="SAPIBP1"  
    Destination="IBP_DESTINATION"  
    TimeAggregationLevel="2"  
    Select="PRDID,LOCID,TSTFR,PERIOD_LEVEL_2,ACTUALSQTY" />  
</IBPReads>
```

## Time Profile

Read time profile data by filling the Key, TypeOfData, TimeProfile, Destination and the Select fields.

### Sample Code

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>  
<IBPReads>  
  <IBPRead Key="TimeProfile"  
    TypeOfData="TIMEPROFILE"  
    TimeProfile="1"  
    Select="PERIODID,DESCR,TSTFR,TSTTO"  
    Destination="IBP_DESTINATION" />  
</IBPReads>
```

## Filtering Data

In the filter expression, you can use the following OData operators and functions:

- Relational operators  
These operators are EQ, NE, LT, LE, GT, GE, and LIKE.
- Logical operators  
These operators are AND, NOT, OR, (, and ).
- Functions  
These functions are STARTSWITH, ENDSWITH, and SUBSTRINGOF.

You can also apply further filters on period levels. To do so, use the following pattern:

#### Sample Code

```
PERIOD_LEVEL_* and PERIOD_LEVEL*_REL
```

#### Sample Code

The following example selects planned independent requirement data related to the `EXAMPLEPRODUCTID` product and has a higher value than 100:

```
FINALDEMANDPLANNINGQTY gt 100 AND PRDID eq 'EXAMPLEPRODUCTID'
```

#### Sample Code

The following example selects planned independent requirement data for the next week or later:

```
PERIOD_LEVEL_2_REL ge 1
```

## 4.1.4 Attachments and Error Logs of the SAP IBP Read - Initialize Reusable Integration Flow

Once the integration flow is complete, you can view the result messages and their attachments in the monitoring tool.

Attachments	Description
1QueryInitInput	Contains the input that the integration flow is getting.
2QueryMergedInput	Contains the input in multi message format.
3AfterInitIBPReadRequestsResults	Contains some attributes and the read request results after the initialization started.
4IBPReadRequestsResults	Contains some attributes and the read request results after the initialization is finished. This is also saved as a header with the <code>IBPReadRequestsResults</code> name.

In case an error occurs, for example, the destination or the master data doesn't exist, the integration flow runs to `Escalated` state. You can see the result on the Monitoring screen or in the attachment of the error message.

## 4.2 SAP IBP Read - Fetch Data

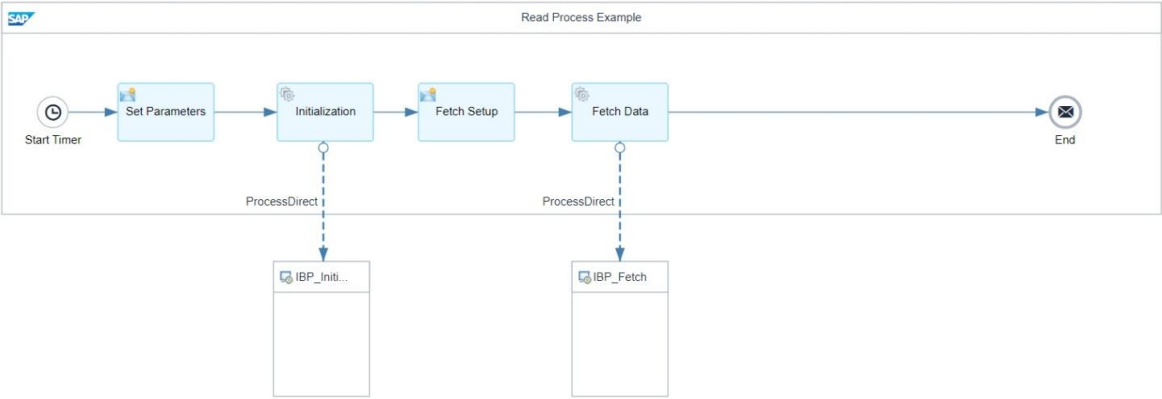
# 4.2.1 Configuring the SAP IBP Read - Fetch Data Reusable Integration Flow

Set the input for the *SAP IBP Read - Fetch Data* reusable integration flow to read master data.

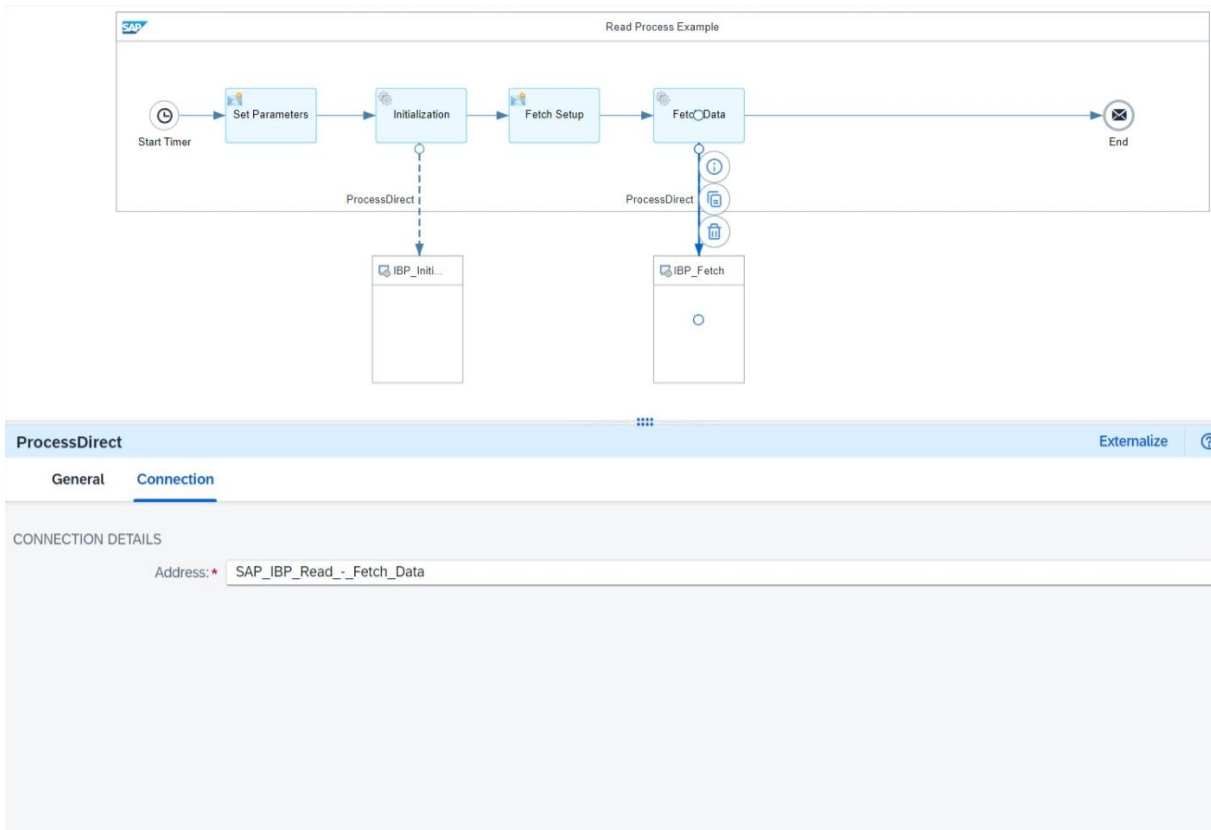
## Prerequisites

You've completed the *SAP IBP Read - Initialize* reusable integration flow process.

## Context



You can call this integration flow with the `SAP_IBP_Read_-_Fetch_Data` process ID.



## Procedure

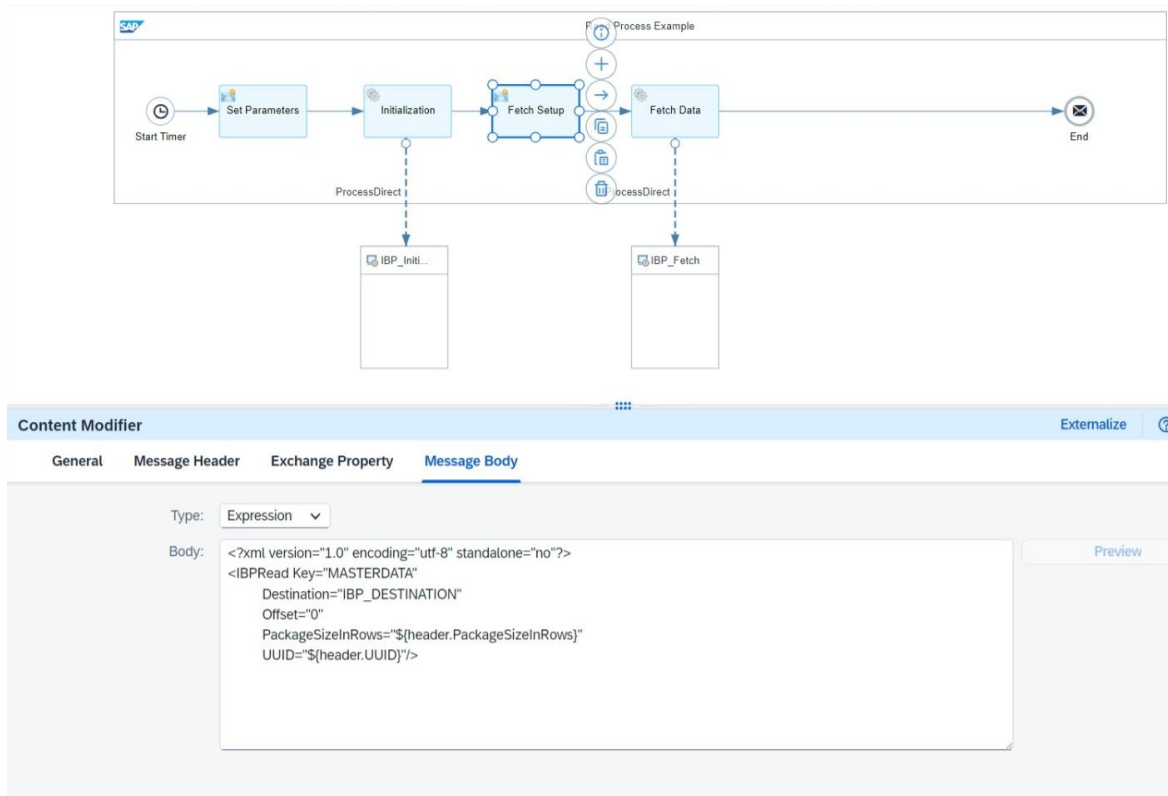
1. Configure the input in the message body of the Content Modifier with the appropriate parameters.

Provide parameters in the body of the *SAP IBP Read - Fetch Data* reusable integration flow in XML format. In this format, you can provide an `IBPRead` node.

Set the input to read master data as follows.

### Sample Code

```
<IBPRead Key="MASTERDATA"
  Offset="0"
  UUID="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  PackageSizeInRows="100"
/>
```



2. Call the integration flow with the Process Direct Call using the ID of the integration flow.

## Results

Once the integration flow is complete, it returns the fetched data in XML format.

### 4.2.2 Message Body Attributes for the SAP IBP Read - Fetch Data Reusable Integration Flow

You can use the following attributes for the `IBPRead` tag when you call the SAP IBP Read - Fetch Data reusable integration flow.

Parameter Name	Description
<code>PackageSizeInRows</code>	<p>You can build a packaging logic around the SAP IBP Read - Fetch Data integration flow parameter, which allows you to specify the desired size of a package in terms of row numbers.</p> <p>Accepted value is in numbers.</p>



Parameter Name	Description
Offset	<p>If you build a packaging logic around the SAP IBP Read - Fetch Data integration flow, the <code>offset</code> parameter can be set to the value that determines where the next package should start fetching the data.</p> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p><b>❖ Example</b></p> <p>You want to read 2000 rows in packages of 1000 rows each. Initially, the offset is set to 0 so the process starts reading from the first record. Once the first package is complete, the <code>offset</code> should be set to <b>1000</b> to ensure that the next package starts fetching from the 1000th record instead of starting from the beginning.</p> </div> <p>Accepted value is in numbers.</p> <p>Default: 0</p>
ParallelThread	<p>If you want to build parallelization around the <a href="#">SAP IBP Read - Fetch Data</a> reusable integration flow, you can set the <code>ParallelThread</code> parameter to identify the threads on the SAP IBP side logs.</p>

### 4.2.3 Attachments and Error Logs of the SAP IBP Read - Fetch Data Reusable Integration Flow

Once the integration flow is complete, you can view the result messages and their attachments in the monitoring tool.

Attachments	Description
FetchMessages	Contains the fetch statuses for all messages.
IBPReads	This value of the header comes from the <a href="#">SAP IBP Read - Initialize</a> reusable integration flow. It contains the <code>IBPReads</code> node with the corresponding input as well as the result of the initialization.
IBPFetchMergedRequests	This contains the input of the integration flow with a message counter.

If the integration flow fails, error messages are created with additional attachments. A custom header is created containing the exception message, destination, keys, and the UUIDs of the read requests.

Exception Messages	Description
ExceptionStackTrace	Contains the exception stack tree.

Exception Messages	Description
Latest message body	Contains the latest message body.

## 4.3 SAP IBP Read - Close and SAP IBP Read - Cancel

### 4.3.1 Configuring the SAP IBP Read - Close and the SAP IBP Read - Cancel Reusable Integration Flow

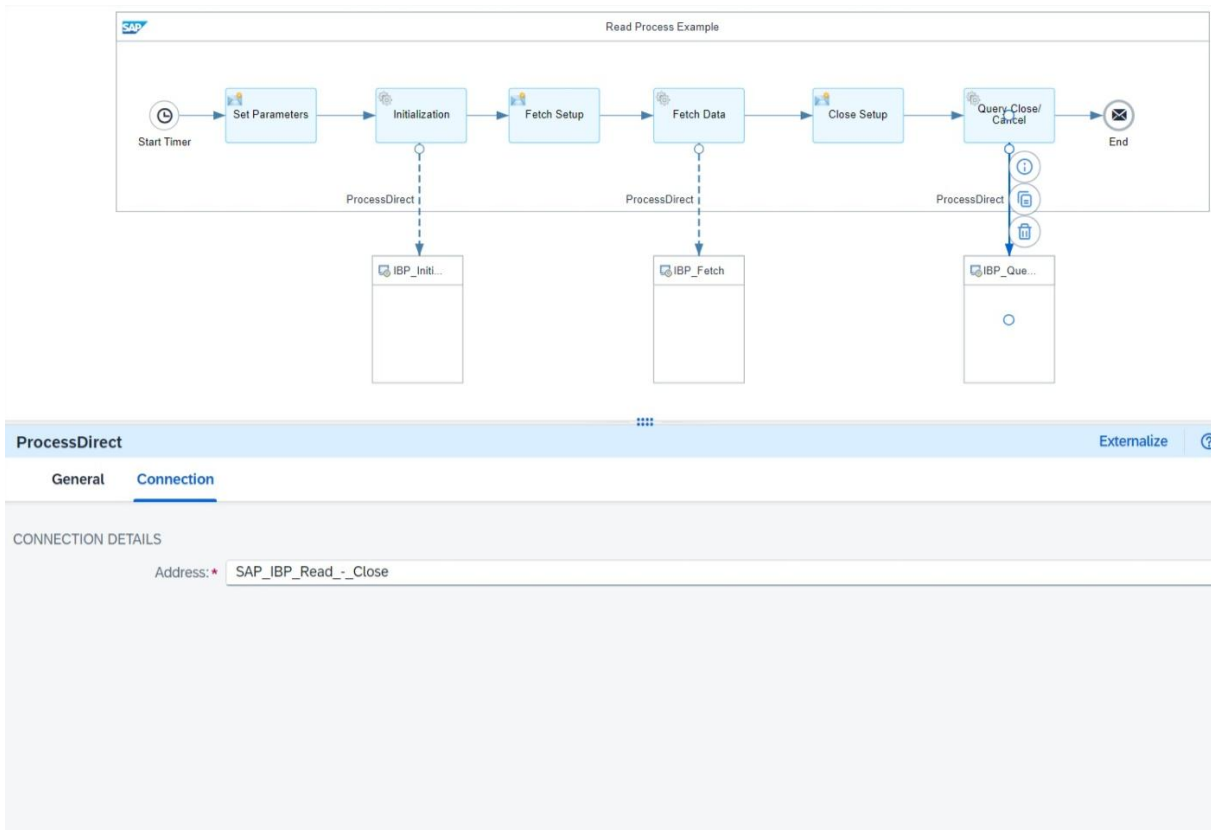
To close queries that have finished successfully, use the [SAP IBP Read - Close](#) reusable integration flow. Call [SAP IBP Read - Cancel](#) integration flow to cancel the query if an error occurs in the calling integration flow that is not caused by the query itself or if you want to stop the query for consistency reasons or to free up resources.

#### Prerequisites

- You've completed configuring the [SAP IBP Read - Fetch Data](#) or the [SAP IBP Read - Fetch Data](#) reusable integration flow.
- You have the ID of the integration flow that you want to close or cancel. You can specify the ID by the integration flow name where the spaces are replaced by underscores. In this example, the process ID is `SAP_IBP_Read_-_Close` or `SAP_IBP_Read_-_Cancel` respectively.

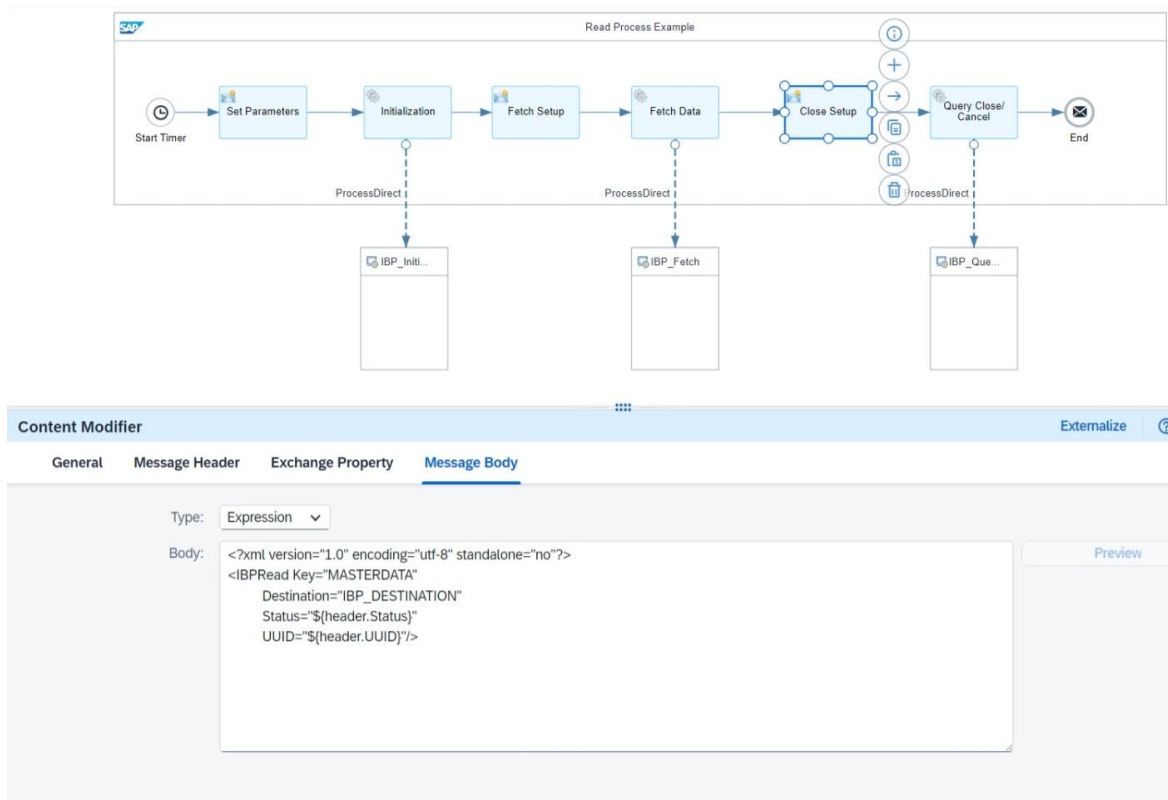
#### Context

Call the SAP IBP Read - Close integration flow using the `SAP_IBP_Read_-_Close` process ID. If you want to call the SAP IBP Read - Cancel integration flow, use the `SAP_IBP_Read_-_Cancel` process ID.



## Procedure

1. To call the *SAP IBP Read - Close* or *SAP IBP Read - Cancel* reusable integration flow, configure the Content Modifier using a Process Direct call.



2. On the *Connection* tab in the Process Direct detail, enter the integration flow ID in the *Address* field.

## Results

If you close the integration flow, the status returns `Finished`. If you've canceled the integration flow, the status returns `Cancelled`. Once the process is complete, the integration flow returns a structure containing the fetched data along with details of the close or cancel process, for example, status, runtimes, and number of calls.

## 4.3.2 Attachments and Error Logs of the SAP IBP Read - Close and the SAP IBP Read - Cancel Reusable Integration Flow

Once the integration flow is complete, you can view the result messages and their attachments in the monitoring tool.

Attachments	Description
<code>IBPReadCloseResults</code>	This attachment is relevant only for the <i>SAP IBP Read - Close</i> reusable integration flow. It includes details such as steps, runtime statuses, and information about the read process. Additionally, it contains the <code>IBPRead</code> header which provides information about the read requests.
<code>IBPReadCancelResults</code>	This attachment is relevant only for the <i>SAP IBP Read - Cancel</i> reusable integration flow. It includes details such as steps, runtime statuses, and information about the read process. Additionally, it contains the <code>IBPRead</code> header which provides information about the read requests.

If the integration flow fails, error messages with additional attachments are created.

## 4.4 Recommendations

Optimize data extraction from SAP IBP through custom integration flows, focusing on the use of filters, splitting data extraction, sequential export scheduling, and enabling traces for monitoring and testing purposes.

- Use filters and optimize the data volume.  
When you define your integration flow for data export from SAP IBP, use filters and optimize the data volume that your custom integration flows extract from SAP IBP. This reduces the runtime of the integration flow and optimizes the resource efficiency, including CPU and memory consumption. By optimizing resource usage, you can minimize the impact on regular business activities and enhance overall system performance.
- Split the data extraction into multiple integration flows.  
Whenever possible, split the data extraction into multiple integration flows, each based on different filter conditions. To optimize resource usage, ensure that only small data sets are extracted from SAP IBP. Use value-based filters for these data sets, and apply time-based filters when extracting key figures.
  - Value-based filter, for example, `"PRDID eq 'PRODUCT1' "`
  - Relative-time-based filter, for example, `"PERIOD_LEVEL_2_REL ge 0"`
  - Absolute-time-based filter, for example, `"PERIOD_LEVEL_2 eq 54321"`
- Schedule the exports to run sequentially. Avoid concurrent execution of the data export integrations.

- We recommend that you enable traces when developing and testing your custom integration flows that call reusable integration flows. In the traces, you can find out how a message is processed and transformed at runtime. For more information, see [Tracing the Execution of an Integration Flow](#).

# 5 SAP IBP Write Reusable Integration Flows

Call the *SAP IBP Write* reusable integration flows in the correct sequence to transfer data from SAP Cloud Integration to SAP IBP.

You can write data to SAP IBP asynchronously:

- Write data to be uploaded into staging tables.
- Once data you want to process in a single transaction is available in staging tables, trigger the post-processing of the data in SAP IBP. Post-processing will validate the data and save valid entries in the core tables to make them available for SAP IBP applications.

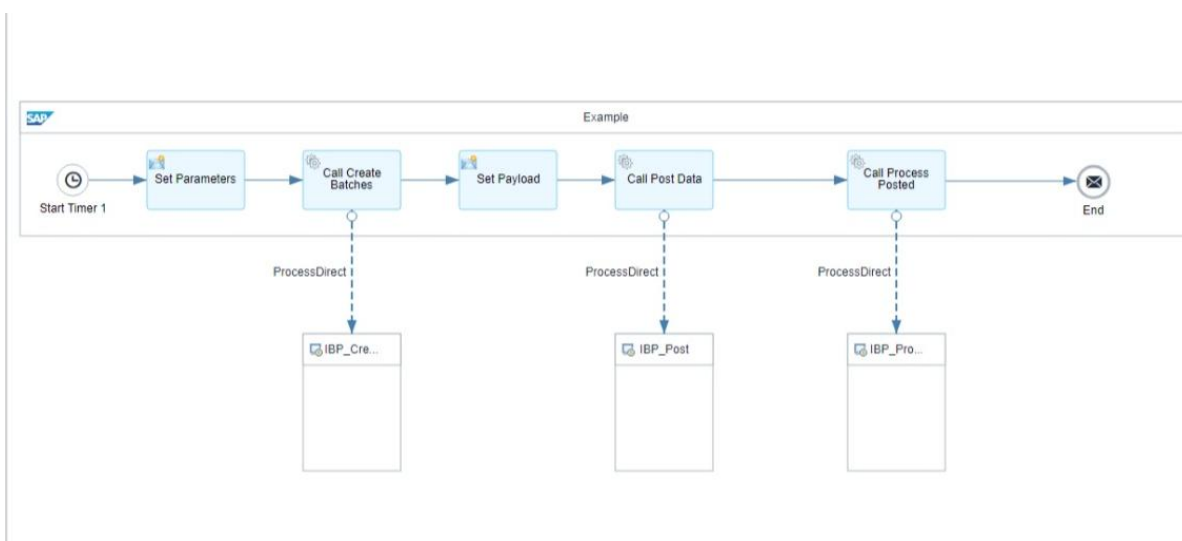
You can upload data to SAP IBP in batches. SAP IBP processes data added to a batch in a single transaction. The batch header defines, for example, the operation type (INSERT\_UPDATE or DELETE) and the planning area and the version you want to upload data.

A batch must include one or more batch items (files), which contain the actual data. Each batch item has its own integration target (for example, a master data type or a planning area for key figures) and a list of columns. For example, if you want to upload only the PRDID and PRDDESC attributes of the product master data type in a batch item, the batch item should have only these two columns.

To upload data to SAP IBP from an integration flow, perform the following steps:

1. Create a batch header by calling *SAP IBP Write - Create Batches*.
2. Create batch items and load data to the staging tables by calling *SAP IBP Write - Post Data*. You can load data into the same batch or batch item with multiple calls to SAP IBP Write - Post Data, in multiple packages if required.
3. Call *SAP IBP Write - Process Posted Data* to start post processing the batch in SAP IBP.

## Results



## 5.1 SAP IBP Write - Create Batches

### 5.1.1 Configuring the SAP IBP Write - Create Batches Reusable Integration Flow

#### Context

To create batches using the SAP IBP Write - Create Batches reusable integration flow, perform the following procedure.

#### → Recommendation

- Specify only one planning area per batch
- If the batch has a key figure upload, the planning area field is mandatory
- Specify only one version for a planning area. If the version field is left empty, data will load to the baseline version
- A batch command must be specified

#### Procedure

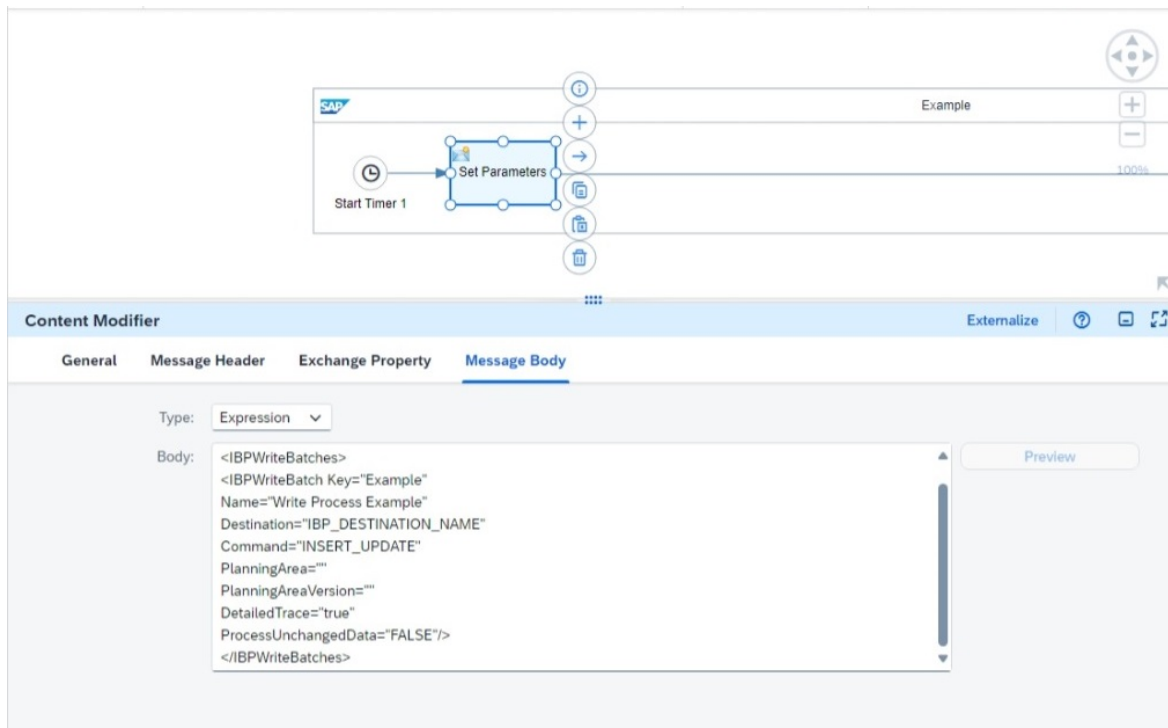
1. Create a batch header by setting the body in the Set Parameters step.

To upload the product master data, use the following example.

#### ↗ Sample Code

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<IBPWriteBatches>
  <IBPWriteBatch
    Key="MasterData"
    Name="Write Master Data"
    Destination="IBP_DESTINATION"
    Command="INSERT_UPDATE"
    PlanningArea=""
    PlanningAreaVersion=""
    DetailedTrace=""
    ProcessUnchangedData="FALSE" />
</IBPWriteBatches>
```





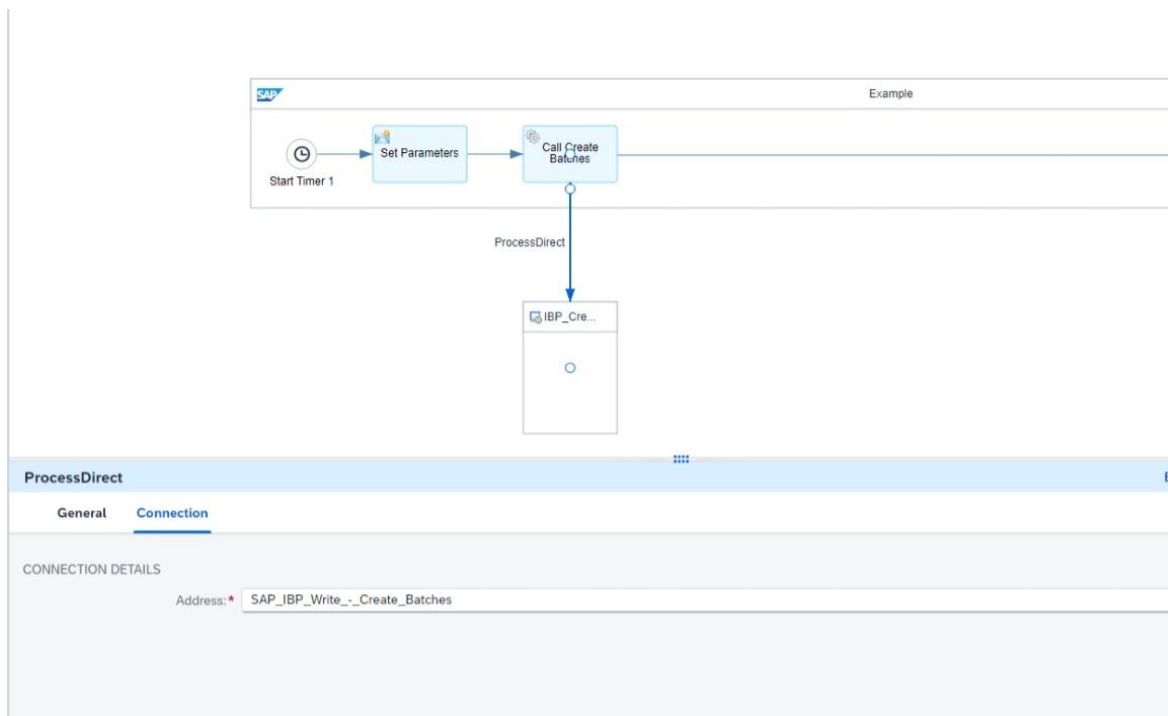
To upload key figure, use the following example.

### Sample Code

```
<IBPWriteBatches>
  <IBPWriteBatch
    Key="KeyFigure"
    Name="Write Key Figure"
    Destination=" IBP_DESTINATION"
    Command=" INSERT_UPDATE "
    PlanningArea=" SAPIBP1 "
    PlanningAreaVersion=" "
    DetailedTrace=" "
    ProcessUnchangedData=" FALSE " />
</IBPWriteBatches>
```

2. Configure the Process Direct Call step.

The ID of the reusable integration flow is the name where the spaces are replaced by underscores, so in this example the address is: SAP\_IBP\_Write\_-\_Create\_Batches.



## Results

Once the integration flow is complete, it returns the IBPWriteBatch parameter structure with a new *Id* parameter added to the list.

```
<?xml version="1.0" encoding="utf-8"?>
<multimap:Messages xmlns:multimap="http://sap.com/xi/XI/SplitAndMerge">
  <multimap:Message1>
    <IBPWriteBatch
      CallerId="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
      Command="INSERT_UPDATE"
      Destination="IBP_DESTINATION"
      Id="XXXXXX"
      Index="1"
      Key="KeyFigure"
      Name="Write Key Figure"
      PlanningArea="SAPIBP1"
    />
  </multimap:Message1>
</multimap:Messages>
```

You can follow the batch ID status in the *Data Integration Jobs* app. After successfully creating the batch with this integration flow, the status turns to *in preparation*.

## 5.1.2 Message Body Attributes for the SAP IBP Write - Create Batches Reusable Integration Flow

You can use the following attributes for the `IBPWriteBatch` tag when you call the SAP IBP Write - Create Batches integration flow.

Parameter	Description
Key	Identifies an <code>IBPWriteBatch</code> .
Name	A name to be displayed in the <i>Data integration Jobs</i> app. It can include header parameters using the expression mode in the content modifier.
Destination	Determines the SAP IBP system what you want to create a batch for. The name is the destination name that you used during the connection setup.
Command	Defines the operation type. Accepted parameters are: <code>INSERT_UPDATE</code> , <code>REPLACE</code> , <code>DELETE</code> . For more information, see <a href="#">Batch Command 'REPLACE'</a> .
PlanningArea	Specifies the planning area where you want to upload the data. This is mandatory if you want to upload key figures or version-specific master data.
PlanningAreaVersion	Specifies the version of the planning area where you want to upload the data. If the master data is not set as version-specific master data in the planning area configuration, it is loaded to the baseline version.
DetailedTrace	By setting the value to <code>true</code> , you can enable additional technical logs on the SAP IBP side for an SAP incident processor.
ProcessUnchangedData	If you want to skip unchanged data, regardless of SAP IBP global parameter setting for this behavior, set this parameter to <code>false</code> . If you want to process unchanged data, regardless of the SAP IBP global parameter setting, set this parameter to <code>true</code> . If you leave this parameter field empty, the SAP IBP global parameter determines the behavior.

## 5.1.3 Writing Various Data Types

By configuring the content modifier, you can write different types of data in SAP IBP as follows:

### Master Data

#### Sample Code

```
?xml version="1.0" encoding="UTF-8" standalone="no"?>
<IBPWriteBatches>
  <IBPWriteBatch
    Key="MasterData"
    Name="Write Master Data"
    Destination="IBP_DESTINATION"
    Command="INSERT_UPDATE"
    PlanningArea=""
    PlanningAreaVersion=""
    DetailedTrace=""
    ProcessUnchangedData="FALSE" />
</IBPWriteBatches>
```

### Key Figure

#### Sample Code

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<IBPWriteBatches>
  <IBPWriteBatch
    Key="KeyFigure"
    Name="Write Key Figure"
    Destination="IBP_DESTINATION"
    Command="INSERT_UPDATE"
    PlanningArea=""
    PlanningAreaVersion=""
    DetailedTrace=""
    ProcessUnchangedData="FALSE" />
</IBPWriteBatches>
```

## 5.1.4 Error Handling

Errors can occur, for example, if you provide invalid planning area or version.

If a batch can't be created, the integration flow returns an error message and the body includes an error statement.

## Output Code

```
<?xml version="1.0" encoding="utf-8"?>
<multimap:Messages xmlns:multimap="http://sap.com/xi/XI/SplitAndMerge">
  <multimap:Message1>
    <IBPWriteBatch
      Command="INSERT_UPDATE"
      Destination="IBP_DESTINATION"
      Id=""
      Index="1"
      Key="ErrorExample"
      Name="TEST RUN"
      PlanningArea="BADPLANNINGAREA"
      PlanningAreaVersion="THISISBADVALUE">
      <CreateMessages>
        <Message Type="Error">
          There was no planning area version THISISBADVALUE found
          with active planning area BADPLANNINGAREA. Input business validation failed.
        </Message>
      </CreateMessages>
    </IBPWriteBatch>
  </multimap:Message1>
</multimap:Messages>
```

You can use the following information to include an error message in the attachment.

## Sample Code

```
def Message processData(Message message) {
  def headers = message.getHeaders();
  def properties = message.getProperties();
  def messageLog = messageLogFactory.getMessageLog(message);
  def bodyAsString = message.getBody(java.lang.String);
  messageLog.addAttachmentAsString('Example error as attachment',
  bodyAsString, 'text/xml')
  return message;
}
```

## 5.2 SAP IBP Write - Post Data

You can use the *SAP IBP Write – Post Data* reusable integration flow to write a set of data to a staging table. You can call the integration flow several times to write data to various batches, and various files (batch item) to add data in packages. Each call of the integration flow adds data to a single file of a single batch at a destination. The values of the `Destination`, `BatchKey`, and `FileName` attributes of the `<IBPWriteMasterData>` or `<IBPWriteKeyFigures>` tag determine destination, batch, and file.

If the file with a value of the `FileName` attribute does not exist yet for the given batch, it will be created with the field list in the `FieldList` attribute. The field list must be the same for all subsequent integration flow calls for the same batch.

You can write master data and key figure data to SAP IBP using the reusable integration flows. Use a single `<IBPWriteMasterData>` tag to write master data entries, or a single `<IBPWriteKeyFigures>` tag to write key figure entries. Both tags can take attributes that define the properties of the file and one or more `<item>` tags that define the actual data entries to write. Inside the `<item>` tag, you can specify the field values of

the given entry using tags with the field names. You can only use field names that you listed in the `FieldList` attribute above.

## Writing Master Data to SAP IBP

You can write data for simple or compound master data types to SAP IBP. If a file is created for a master data type as a target, the field list must be the same for all subsequent integration flow calls for the master data type.

The field list must include all key attributes of the master data type and each item must have a value for each of those attributes. When deleting data, that is, the command attribute for the batch is **DELETE**, all non-key attributes are ignored. When inserting or updating data, that is, the command attribute for the batch is **INSERT\_UPDATE**, only those master data attributes are changed that are listed in the value of the `FieldList` attribute. Other attributes are skipped for existing entries and are set to **NULL** for new entries.

To update a non-key attribute of existing records to **NULL**, you must add the field to the value of the `FieldList` parameter and then either skip specifying the corresponding tag within the `<item>` tag or specify the tag with the XML constant `nil` as a value.

Use the [Master Data Types – Model Configuration](#) app in SAP IBP to see the list of key and non-key attributes for a master data type.

## Writing Key Figure Data to SAP IBP

You can write data to multiple key figures in a single file, but each of them must be a stored key figure on the same planning level of the planning area that you earlier specified for the batch. If a file is created to write key figures, the field list must be the same for all subsequent integration flow calls for the same key figures on the same planning level.

The field list must include all non-time root attributes of the planning level and each item must have a value for each of those attributes. If the planning level is time-dependent, the field list must also include a `KEYFIGUREDATE` field. The value of a `<KEYFIGUREDATE>` tag within an `<item>` tag will be a date within the corresponding SAP IBP time period, specified using the YYYY-MM-DD format. The time period is determined by the value of the `TimeDisaggregationLevel` attribute of the `<IBPWriteKeyFigures>` tag. Use time level values that are defined in the time profile of the planning area using the [Planning Areas – Model Configuration](#) app. If you choose, for example, the time level value for month, the value of `<KEYFIGUREDATE>` should be a date within that month. You can only specify a `TimeDisaggregationLevel` value that is the same or higher than the base time profile level of the planning level you're uploading data to. If `TimeDisaggregationLevel` is left empty, it defaults to the base time profile level. Have a single data entry for each root attribute value and time period combination across each integration flow call for the same file to avoid non-uniqueness errors.

To update a key figure for existing records to **NULL**, you must add the key figure name to the value of the `FieldList` parameter and then perform one of the following steps

- skip specifying the corresponding tag withing the `<item>` tag
- specify the attribute `xsi:nil` as **true** for the tag, for example, `<ABCDESCR xsi:nil="true" />`
- specify an empty value without whitespace to the tag, for example `<ABCDESCR></ABCDESCR>`

To set a non-key attribute to an empty string, specify a single space character as the value for the tag, for example, <ABCDESCR> </ABCDESCR>.

Use the *Planning Areas – Model Configuration* app to see the list of root attributes, key figures, and base time profile level for a planning level of a planning area.

## 5.2.1 Configuring the SAP IBP Write - Post Data Reusable Integration Flow

### Prerequisites

You have completed the *SAP IBP Write - Create Batches* process.

### Procedure

1. Create batch items and load the data into the staging table.

The post data input can have a `DataFormat` attribute. The value of the data format can be JSON or XML. If you don't specify the data format, it is interpreted as XML. Perform one of the following options:

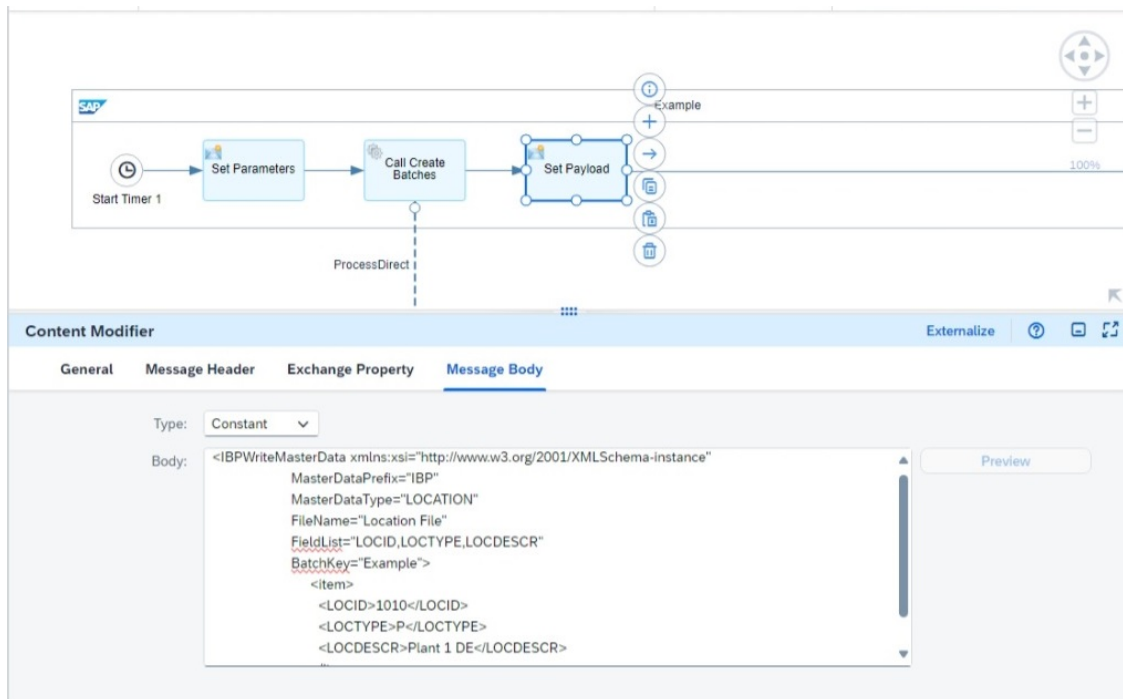
- a. **Use XML Data Format**

#### Master Data

See the following example to define the `IBPWriteMasterData` payload. This payload uploads a single line of data to the staging table of the master data type `TI4LOCATION`. The values of the fields `LOCID`, `LOCTYPE`, and `LOCDESCR` get the values of the corresponding tags. The value `LOCREGION` is in the `FieldList` attribute, but does not get a value in the `<item>` tag, so it will be set to `NULL`.

#### Sample Code

```
<IBPWriteMasterData xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    MasterDataPrefix="IBP"
    MasterDataType="LOCATION"
    FileName="Location File"
    FieldList="LOCID,LOCTYPE,LOCDESCR"
    BatchKey="Example" >
  <item>
    <LOCID>1010</LOCID>
    <LOCTYPE>P</LOCTYPE>
    <LOCDESCR>Plant 1 DE</LOCDESCR>
  </item>
</IBPWriteMasterData>
```



## Key Figures

If you want to write key figures, define the file name and the payload. This payload uploads a single value to the key figure ACTUALSQTY for the PRDID-LOCID-CUSTID root attribute combination as specified and for the technical week that includes the date June 1, 2023 as follows:

### Sample Code

```

<IBPWriteKeyFigures xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  FieldList="PRDID,LOCID,CUSTID,KEYFIGUREDATE,ADJDELIVQTY"
  BatchKey="Example"
  FileName="Key Figure File"
  TimeDisaggregationLevel="1">
  <item>
    <PRDID>IBP-100</PRDID>
    <LOCID>1010</LOCID>
    <CUSTID>DUMMY</CUSTID>
    <KEYFIGUREDATE>2024-11-01</KEYFIGUREDATE>
    <ADJDELIVQTY>1</ADJDELIVQTY>
  </item>
</IBPWriteKeyFigures>

```

## b. Use JSON Data Format

### Master Data

#### Sample Code

```

<IBPWriteMasterData xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  MasterDataPrefix="IBP"
  FileName=" Location File "
  MasterDataType="LOCATION "

```



```

FieldList=" LOCID,LOCTYPE,LOCDESCR "
BatchKey="Example"
DataFormat="JSON">
<item>
<JSON_DATA_STRING>
{
  "ITEMS": [
    {
      "LOCID": "1010",
      "LOCTYPE": "P",
      "LOCDESCR": "Plant 1 DE"
    },
    {
      "LOCID": "1020",
      "LOCTYPE": "P",
      "LOCDESCR": "Plant 2 DE"
    }
  ]
}
</JSON_DATA_STRING>
</item>
</IBPWriteMasterData>

```

## Key Figure

### { } Sample Code

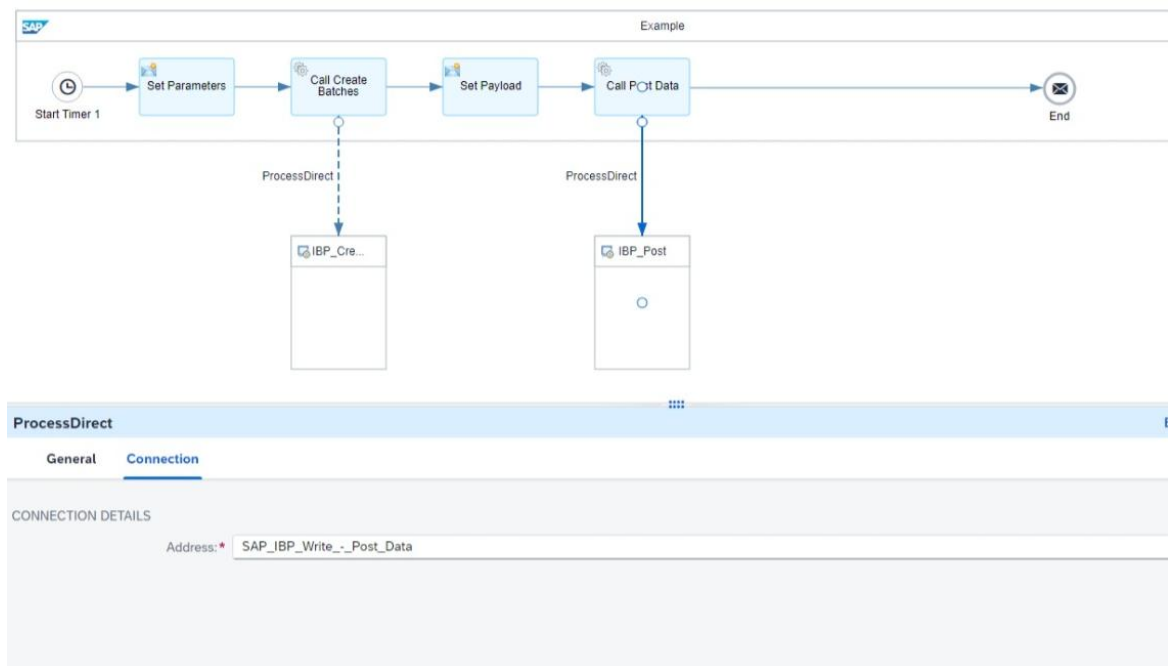
```

<IBPWriteKeyFigures xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
                    FileName=" Location File "
                    FieldList="
PRDID,LOCID,CUSTID,KEYFIGUREDATE,ADJDELIVQTY
                    BatchKey="Example"
                    TimeDisaggregationLevel="1"
                    DataFormat="JSON">
<item>
<JSON_DATA_STRING>
{
  "ITEMS": [
    {
      "PRDID": "IBP-100",
      "LOCID": "1010",
      "CUSTID": "DUMMY"
      "KEYFIGUREDATE": "2024-11-01"
      "ADJDELIVQTY": "1"
    }
  ]
}
</JSON_DATA_STRING>
</item>
</IBPWriteKeyFigures>

```

2. Configure the Process Direct Call step.

The ID of the reusable integration flow is the name where the spaces are replaced by underscores, so in this example the address is: SAP\_IBP\_Write\_-\_Post\_Data.



## 5.2.2 Message Body Attributes for the SAP IBP Write - Post Data Reusable Integration Flow

You can use the following attributes for the `IBPWriteMasterData` and the `IBPWriteKeyFigure` tag when you call the *SAP IBP Write - Post Data* reusable integration flow.

Parameter Name	Description
<code>MasterDataPrefix</code>	The prefix that you use during the planning area creation.
<code>MasterDataType</code>	Determines the master data type without the prefix. This attribute is mandatory when writing master data entries using the tag <code>IBPWriteMasterData</code> . The master data type name that you defined in the SAP IBP model concatenates both values of the <code>MasterDataPrefix</code> and the <code>MasterDataType</code> .
<code>Destination</code>	Determines the SAP IBP system where the batch will be created.
<code>FileName</code>	Defines the file name.
<code>FieldList</code>	A list of attributes to be uploaded, separated by commas.

Parameter Name	Description
BatchKey	The key is defined in the payload before calling the Create Batches reusable integration flow.
TimeDisaggregationLevel	Determines what time level you are loading the key figures.
DataFormat	<p>Determines the data format. Accepted values: XML or JSON (case insensitive). This attribute is not mandatory.</p> <ul style="list-style-type: none"> <li>• If the attribute is set to XML, the incoming data is processed as XML format.</li> <li>• If the attribute is set to JSON, the incoming data is processed as JSON format.</li> <li>• If the value is empty, the incoming data is processed as XML format by default.</li> <li>• If the attribute contains a value other than XML or JSON, the integration flow triggers an escalation.</li> </ul>

## 5.3 SAP IBP Write - Process Posted Data

### 5.3.1 Configuring SAP Write - Process Posted Data Reusable Integration Flow

#### Prerequisites

You've created the batches and uploaded the data to the staging tables.

#### Procedure

Call the [SAP Write - Process Posted Data](#) integration flow with your batch ID in the body.

## Results

If the data upload is successful, it returns the payload as follows:

### Sample Code

```
<IBPWriteBatch Command="INSERT_UPDATE"
  CreatedAt="2024-11-01T00:00:00Z"
  CreatedBy="CI_COM_USER"
  Destination="IBP_DESTINATION"
  Id="XXXXXX"
  Key="Example"
  Name="Location File"
  ProcessingEndedAt="2024-11-01T00:00:30Z"
  ProcessingStartedAt="2024-11-01T00:00:20Z"
  ScheduledAt="2024-11-01T00:00:10Z"
  WorstProcessingStatus="PROCESSED">
<Files>
<File Count="1"
  ErrorCount="0"
  MasterDataType="IBPLOCATION"
  Name="Location File"
  Status="PROCESSED"
  TypeOfData="Master Data"/>
</Files>
```

In this example, after completion, the IBPWriteBatch is extended with the following information properties:

Parameter Name	Description
ScheduledAt	The time when IBP got informed to start the post processing.
ProcessingStartedAt	The time when the processing started.
ProcessingEndedAt	The time when the processing finished completely.
WorstProcessingStatus	The first status among all the files, possible statuses are, for example, PROCESSED, PROCESSED_WITH_ERRORS, ERROR.
Count	The number of rows being uploaded.
MasterDataType	The type of master data.
Name	The file name.
Status	The status of that specific file.
TypeOfData	The data type, for example, master data, or key figure.

## 5.3.2 Error Handling

If an error occurs, the payload returns a processed with errors status with the following parameters:

Parameter	Description
ErrorCount	Shows the number of failed rows.
FirstError	Specifies the first error among the failed rows. For more details, check the Data Integration Jobs app.
FirstErrorId	Defines the ID of the first error message.
FirstErrorTitle	The group title of the first error.

## 5.4 Recommendations

When integrating data to SAP Integrated Business Planning, you can manage and limit data volume to ensure optimal performance and efficiency as follows:



- Assess data requirement: evaluate the necessity of each data element before including it in the integration run. Only include data that is essential for the planning process.
- Data filtering: apply filters to exclude unnecessary data. For example, filter out historical data that is not relevant to the current planning cycle.
- Incremental data loads: use incremental data loads to transfer only the data that has changed since the last integration run. This approach minimizes the data volume and reduces processing time.
- Monitor and optimize: continuously monitor the performance of your integration runs and optimize the data volume based on the insights gained. Adjust filters, aggregation levels, and other parameters as needed.

# Important Disclaimers and Legal Information

## Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.

About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

## Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

## Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.

The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

## Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

## Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.



© 2025 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see <https://www.sap.com/about/legal/trademark.html> for additional trademark information and notices.