

Integration Guide | PUBLIC
Document Version: 1.1.3 – 2026-09-04

SAP IBP Add-On - Reusable Integration Flows

Data Integration Between SAP ERP or SAP S/4HANA Supply Chain Integration Add-On and SAP Integrated Business Planning through SAP Cloud Integration using SAP IBP Add-On - Reusable Integration Flows

© 2026 SAP SE or an SAP affiliate company. All rights reserved.

Content

1 Document History. . . . 3

2 Introduction. . . . 4

3 Prerequisites. . . . 5

4 SAP IBP Add-On Read - Generic Process Flow. . . . 6

4.1 Stages of the Integration Flow. . . . 6

4.2 Integrating Data into SAP IBP. . . . 6

5 Configuring the Integration Flow. . . . 8

5.1 Message Body Attributes for the SAP IBP Add-On Read - Generic Process Flow. . . . 10

5.2 ODP 2.0 and ABAP CDS View Integration. . . . 12

5.3 Data Mapping. . . . 13

5.4 Time Aggregation During Key Figure Uploads. . . . 14

5.5 Filtering Configuration. . . . 15

5.6 Working with Extensions. . . . 16

Post Fetch Extension. . . . 16

Custom Mapping Extension. . . . 19

6 Troubleshooting and Error Handling. . . . 20

7 Recommendations. . . . 22

1 Document History

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

Version	Date	Description
1.1.3	Sept 4, 2026	Added `SubscriberName` to Message Body Parameters in section Message Body Attributes for the SAP IBP Add-On Read - Generic Process Flow [page 10].
1.1.2	Aug 1, 2026	Updated document with information about ODP 2.0 protocol support and ABAP CDS (Core Data Services) view integration.
1.1.1	May 3, 2026	Corrected the maximum package size to be defined in Bytes.
1.1.0	November 26, 2025	Updated document with minor refinements.
1.0.0	August 29, 2025	Initial version

2 Introduction

You can use the *SAP IBP Add-On - Reusable Integration Flows* to transfer data between the SAP ERP or SAP S/4HANA supply chain integration add-on (add-on) and SAP Integrated Business Planning (SAP IBP).

Data integration through SAP Cloud Integration using the artifacts in the *SAP IBP Add-On – Reusable Integration Flows* package is available as of 2508.

To transfer data into SAP IBP, use the *SAP IBP Add-On Read - Generic Process Flow*.

3 Prerequisites

To transfer data, you must configure the connection between SAP IBP and SAP Cloud Integration as follows:

- Enable the *Planning - Integration Suite-Cloud Integration Integration* (`SAP_COM_0931`) communication scenario that allows a connection between SAP IBP and SAP Cloud Integration. For more information, see Integrating Data Using SAP Cloud Integration.
- Install and configure your add-on. For more information, see Administrator's Guide - SAP ERP, Supply Chain Integration Add-On for SAP Integrated Business Planning 1.1 or Administrator's Guide - SAP S/4HANA, Supply Chain Integration Add-On for SAP Integrated Business Planning.
- Install and configure your SAP Cloud Connector and expose your add-on to be available for SAP Cloud Integration.
- Configure the SAP BTP / SAP Integration Suite to connect to SAP IBP using the communication arrangement, and to the add-on using your SAP Cloud Connector. For more information, see 3628813.
- Deploy the *SAP IBP Add-On – Reusable Integration Flows* package on SAP Cloud Integration.

4 SAP IBP Add-On Read - Generic Process Flow

To read data from the add-on extractors and to transfer data into SAP IBP, you can use the *SAP IBP Add-On Read – Generic Process Flow*.

4.1 Stages of the Integration Flow

This integration flow consists of multiple steps and goes through the following stages:

1. Set and Validate Input Parameters as Preparation
   The available parameters are described in Message Body Attributes for the SAP IBP Add-On Read - Generic Process Flow [page 10].
2. Create Batches
   In this step, the flow creates the batch header data in SAP IBP based on the parameters you specified.
3. Transfer Data
   During data transfer, the integration flow calls the specified add-on extractor via RFC, fetches the data based on the defined mapping and filtering, and posts it to the staging tables in SAP IBP.
4. Process Posted Data
   Once the data transfer is complete, the post processing starts to validate and process the data in SAP IBP.

4.2 Integrating Data into SAP IBP

Integrating data to SAP IBP is asynchronous. First, data is uploaded into staging tables. Once the data you want to process in a single transaction is available in the staging tables, the post processing of the data is triggered in SAP IBP. Post processing validates the data and saves valid entries in the core tables to make them available for SAP IBP applications.

Uploading Data to SAP IBP in Batches

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 into.

A batch must include one or more batch items (files) that contain the 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 include only these two columns.

5 Configuring the Integration Flow

By following this procedure, you can create batches, read and fetch data, and call the post processing using the integration flow.

→ Recommendation

- Create a new wrapper integration flow around the delivered generic process flow for better and easier configuration.
- Parameters should be defined as *Message Body*. If you define the parameters in the *Message Header*, you must add another step in the flow to convert the header parameters into message body.
- Specify only one planning area per batch.
- The following fields are mandatory:
  - `DestinationforAddon`
  - `DestinationforSAPIBP`
  - `PlanningArea`
  - `DataSource`
  - `BatchCommand`
  - `BatchName`
  - `TypeOfData`
  - `TargetFieldsInSAPIBP`
  - `BatchKey`
- If the batch has a key figure upload, the `TargetTimeProfileLevelinSAPIBP` field is mandatory.
- If the batch has a master data upload, the `MasterDataPrefix`, and `MasterDataType` fields are mandatory.
- Specify only one version for a planning area. If the version field is left empty, data is loaded to the baseline version.

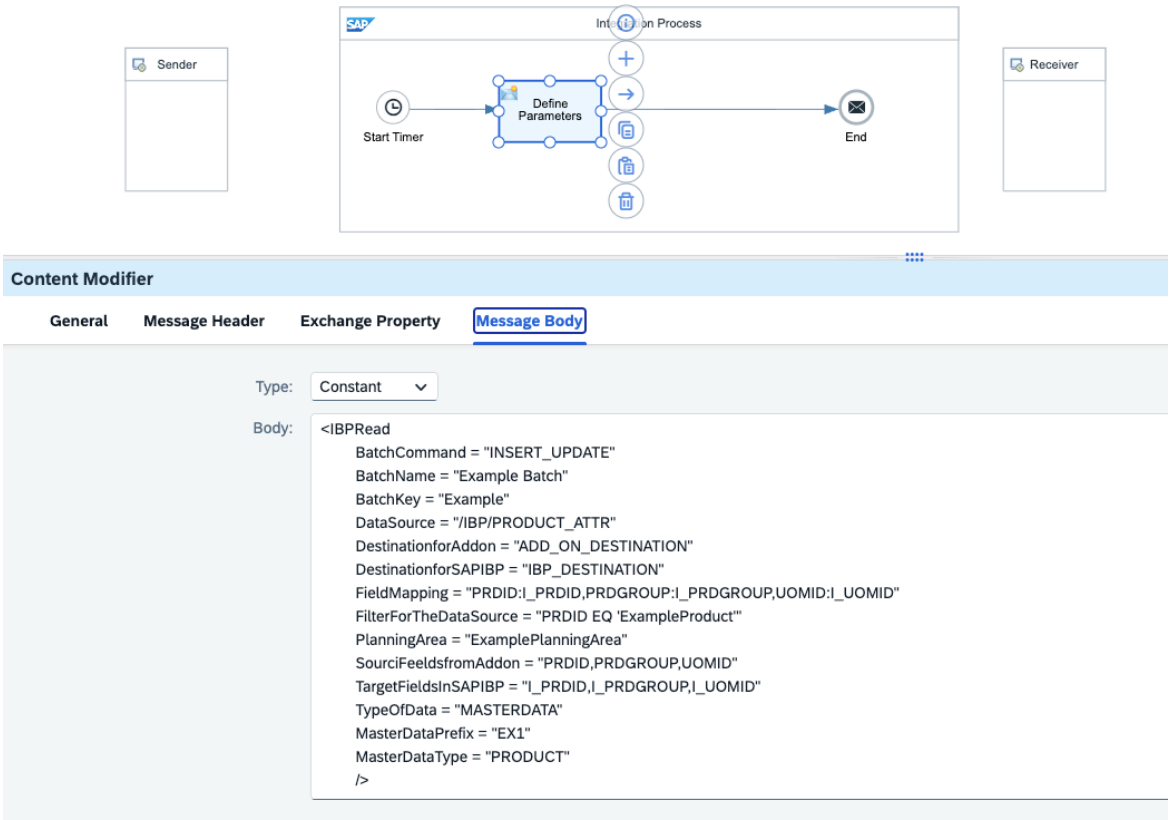
Procedure

1. Create your own wrapper integration flow.
2. Set the parameters in a *Content Modifier* step in the newly created integration flow. For a complete list of parameters that can be used, please see Message Body Attributes for the SAP IBP Add-On Read - Generic Process Flow [page 10].
   To upload product master data, you can use the following example:

   {·} Sample Code

   ```
   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
   <IBPRead
       BatchCommand = "INSERT_UPDATE"
       BatchName = "Example Batch"
       BatchKey = "Example"
   ```

```
    DataSource = "/IBP/PRODUCT_ATTR"
    DestinationforAddon = "ADD_ON_DESTINATION"
    DestinationforSAPIBP = "IBP_DESTINATION"
    FieldMapping = "PRDID:I_PRDID,PRDGROUP:I_PRDGROUP,UOMID:I_UOMID"
    FilterForTheDataSource = "PRDID EQ 'ExampleProduct'"
    PlanningArea = "ExamplePlanningArea"
    SourceFieldsfromAddon = "PRDID,PRDGROUP,UOMID"
    TargetFieldsInSAPIBP = "I_PRDID,I_PRDGROUP,I_UOMID"
    TypeOfData = "MASTERDATA"
    MasterDataPrefix = "EX1"
    MasterDataType = "PRODUCT"
    />
```



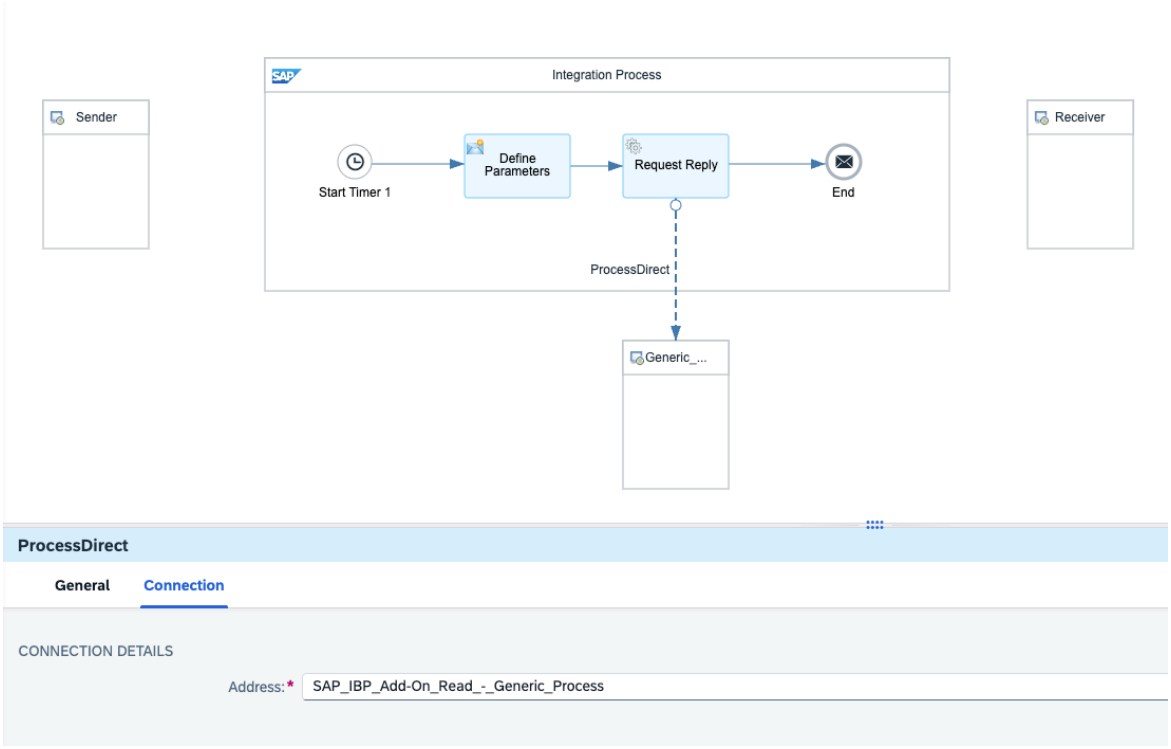
To upload exchange rates key figures, you can use the following example:

{·} Sample Code

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<IBPRead
    BatchCommand = "INSERT_UPDATE"
    BatchName = "Example Batch"
    BatchKey = "ExampleUpload"
    DataSource = "/IBP/EXCHANGE_RATES_KF"
    DestinationforAddon = "ADD_ON_DESTINATION"
    DestinationforSAPIBP = "IBP_DESTINATION"
    FieldMapping =
"CURRENCYID:I_CURRID,CURRENCYTO:I_CURRTOID,EXCHANGERATE:I_EXCHANGERATE,KEYF
IGUREDATE:KEYFIGUREDATE"
    FileName = "ExampleFile"
    MaxPackageSize = "40"
    PlanningArea = "ExamplePlanningArea"
    SourceFieldsfromAddon =
"CURRENCYID,CURRENCYTO,EXCHANGERATE,KEYFIGUREDATE"
```

```
    SourceTimeProfilefromAddon = "0001"
    SourceTimeProfileLevelfromAddon = "2"
    TargetFieldsInSAPIBP =
"I_CURRID,I_CURRTOID,I_EXCHANGERATE,KEYFIGUREDATE"
    TargetTimeProfileLevelinSAPIBP = "2"
    TypeOfData = "KEYFIGURES"
    />
```

3. Configure a *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_Add-On_Read_-_Generic_Process`.



Result

Once the integration flow is complete, it can be saved and deployed, so it runs with the specified parameters.

You can track the status in SAP Cloud Integration or in SAP IBP in the *Data Integration Jobs* app.

5.1 Message Body Attributes for the SAP IBP Add-On Read - Generic Process Flow

You can use the following attributes for the `IBPRead` tag when you call the *SAP IBP Add-On Read – Generic Process* integration flow. You can also find two *XML Schema Definition* files among the *Resources* of the integration flow, one for key figure and one for master data integration. The `IBPRead` *XML Schema Definitions*

describe the expected structure of message bodies for the *SAP IBP Add-On Read - Generic Process Flow*. You can use these schema files to ensure that incoming XML messages conform to the required format for master data and key figure integration. Example XML messages can be found in Configuring the Integration Flow [page 8].

Parameter ID	Description
`BatchCommand`	Defines the operation type. Accepted values are: `INSERT_UPDATE, REPLACE, DELETE`.
`BatchKey`	Identifies an `IBPRead` tag.
`BatchName`	A name to be displayed in the *Data Integration Jobs* app. It can include header parameters using the expression mode in the content modifier.
`CustomMappingExtensionIntegrationFlowAddress`	Specifies the integration flow that you have created to use the *Custom Mapping Extension*.
`DataSource`	Determines the add-on extractor or the ABAP CDS view that you want to use.
`DestinationforAddon`	Determines the source system from where you want to read the data.
`DestinationforSAPIBP`	Determines the SAP IBP system where you want to load the data.
`FieldMapping`	Specifies the field mapping between the add-on Extractor fields and the target SAP IBP fields.
`FileName`	Specifies the file name.
`FilterForTheDataSource`	Specifies the filter criteria you want to use while reading the data from the add-on.
`MasterDataType`	Determines the master data type without the prefix. This attribute is mandatory when writing master data entries. The master data type name that you defined in the SAP IBP model concatenates both values of the `MasterDataPrefix` and the `MasterDataType`.
`MasterDataPrefix`	The prefix that you set during the planning area creation.
`MaxPackageSize`	Defines the maximum package size in Bytes. For more information about package support, see SAP Note 2196500.
`ODPVersion`	If you want to use ABAP CDS views, set this parameter to **`2.0`**.

Parameter ID	Description
`ParallelDataTransfer`	Determines if parallel data transfer should happen.
`PlanningArea`	Specifies the planning area where you want to upload the 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.
`PostFetchFilterExtensionAddress`	Specifies the integration flow you have created to use the *Custom Mapping Extension*.
`ProcessUnchangedData`	You can control whether unchanged data is processed using the following options: • To skip unchanged data, regardless of the SAP IBP global parameter setting for this behavior, set this parameter to *false*. • To process unchanged data, regardless of the SAP IBP global parameter setting, set this parameter to *true*. • If you leave this parameter field empty, the SAP IBP global parameter determines the behavior.
`SourceFieldsfromAddon`	Specifies the requested source fields from the add-on.
`SubscriberName`	Specifies the business object name (or subscriber identification) passed to the add-on during the ODP RFC call. This parameter identifies the caller context in the source system to comply with SAP ODP security restrictions, see SAP Note 3748819.
`TargetFieldsInSAPIBP`	Specifies the target fields in SAP IBP.
`TargetTimeProfileLevelinSAPIBP`	Define the target time profile level in SAP IBP for disaggregation.
`TypeOfData`	Specifies the type of the data, which can either be `MASTERDATA` or `KEYFIGURES`.

5.2 ODP 2.0 and ABAP CDS View Integration

You can use ODP 2.0 protocol support and ABAP CDS (Core Data Services) view integration to transfer data to SAP Integrated Business Planning (SAP IBP). In modern SAP landscapes, particularly SAP S/4HANA, ABAP CDS views serve as the primary data source. They provide a high-performance extraction layer that uses the ODP 2.0 framework for data transfer to SAP IBP.

Creation and ODP Enablement of ABAP CDS Views

Unlike classic DataSources, ABAP CDS views are developed exclusively within the ABAP Development Tools (ADT) in Eclipse. To make a CDS view available for ODP-based extraction, include the following annotations in the DDL (Data Definition Language) source:

- `@Analytics.dataExtraction.enabled: true` - This is the core annotation that registers the CDS view within the ODP framework.
- `@Analytics.dataCategory` - Defines the nature of the data (for example, **`#FACT`** for transactional data or **`#DIMENSION`** for master data attributes).

Following the ODP naming convention, the DataSource name is derived from the CDS view entity name followed by a category suffix (for example, `Z_SALES_DATA$F`). The dollar sign (`$`) is a critical technical indicator that integration flows use to identify modern CDS-based sources.

Integration Flow Enhancements and Branching Logic

The *Generic Data Transfer* integration flow (iFlow) has been refactored to include a hybrid branching logic. This allows a single integration interface to support both legacy ODP 1.0 (SAPI) environments and modern ODP 2.0 (SAPI or ABAP CDS) systems dynamically:

- **Dynamic versioning:** The `IBPRead` input XML now supports an `ODPVersion` attribute. When set to **`2.0`**, the iFlow bypasses legacy modules and invokes the `RODPS_REPL_ODP_...` series of RFCs.
- **Automated context determination:** XSLT mappings have been enhanced to inspect the DataSource name. If the name contains a `$`, the `I_CONTEXT` parameter is automatically set to **`ABAP_CDS`**. If absent, the system defaults to **`SAPI`** to maintain backward compatibility with classic ERP extractors.
- **XSTRING payload handling:** In ODP 2.0, data is transmitted as a compressed binary XSTRING. The updated Groovy scripts within the iFlow handle this Base64-encoded payload. By processing the entire record as a single binary block, the iFlow eliminates the legacy 255-character limit and the associated continuation logic, reducing memory consumption and increasing throughput on the SAP Cloud Integration platform.

5.3 Data Mapping

Mapping the add-on extractor fields to the SAP IBP fields can be done in the *Message Body*.

The following key parameters of the integration flow are needed for mapping:

- `SourceFieldsfromAddon`
  It must contain the requested fields from the *Extractor* separated by a comma.
- `TargetFieldsInSAPIBP`
  It must contain the target fields in SAP IBP separated by a comma.
- `FieldMapping`
  It must contain the field mappings between the source fields and the target fields, in `SOURCE:TARGET` format.

For example, `PRDID:I_PRDID,LOCID:I_LOCID,ACTUALSQUANTITY:I_ACTUALSQTY.`
You can map constant values in the mapping between single quotes.
For example, `'DUMMY':I_TSPCUSTID`.
You can't use quotation marks in the constant value.

5.4 Time Aggregation During Key Figure Uploads

You can configure a time profile level in SAP IBP and upload the SAP IBP time profile in the add-on. By synchronizing the time profile levels, you can process data without aggregation attempts.

You can configure the parameter for the target time profile level in SAP IBP (see the table in Message Body Attributes for the SAP IBP Add-On Read - Generic Process Flow [page 10]) and use the source time profile ID and source time profile level in the source system, defined as filter criteria. These parameters can be combined in different ways.

In the integration flow, you can use the time profile that you have uploaded to the add-on as filter criteria. That means, you have to specify them in the *Message Body*, as `FilterForTheDataSource` parameter values.

> ✤ Example
>
> `FilterForTheDataSource = "TIMEPROFILEID EQ '0001',TIMEPROFILELEVEL EQ '04'"`

SAP IBP can integrate key figure data on the base time profile level or disaggregate the key figure data if it's uploaded on a higher time profile level. To ensure success, data must be loaded on a time profile level that is not below the base time profile level of the key figure.

Example

If the following conditions are fulfilled, the data will be disaggregated on the key figure's base time profile level:

- The key figure configuration allows for disaggregation.
- The time profile level *4* means month in SAP IBP.
- The `TargetTimeProfileLevelinSAPIBP` is set to *4*.
  This indicates that you want to load the data on a monthly level.
- The base time profile level of the loaded key figure is the technical week.
- In the `FilterForTheDataSource` parameter you also filter for `TIMEPROFILELEVEL EQ '4'` (monthly level).

5.5 Filtering Configuration

Use a simplified filter to narrow down the data load you want to integrate. You can filter for individual entries or ranges of entries, or entries in a specific pattern. For more complex filters, use the `PostFetchExtension`, where you can define custom filtering in a Groovy script file.

As the filter conditions are translated into ABAP selection tables in the background, there are a few rules and limitations that must be followed.

Using the `FilterForTheDataSource` parameter for individual entries, you can add the field and the exact values you want the filter for. The filtered values must always be put between a pair of single quotation marks, and each filter condition must be separated by a comma.

The following operators are supported by the integration flow: `EQ, NE, CP, NP, BT, NT`.

Example

In these examples, you can see the results of the different filtering techniques:

- If you filter for **`ExampleProduct1`**, only data that contains ExampleProduct1 is integrated.
  `FilterForTheDataSource` = **`"PRDID EQ 'ExampleProduct1'"`**
- You can filter for different values for different fields.
  `FilterForTheDataSource` = **`"PRDID EQ 'ExampleProduct1', LOCID EQ 'ExampleLocation1'"`**
- You can filter for multiple values of the same field. The values should be written into simple conditions or you can specify them as a range.
  `FilterForTheDataSource` = **`"PRDID BT 'ExampleProduct1' 'ExampleProduct3'"`**
- You can filter for multiple values of multiple fields, similarly to the example above, the conditions should be written one after another.
- You can filter records based on the condition that a field contains a specific substring.
  `FilterForTheDataSource` = **`"PRDID CP 'EXAMPLE_PHONE_*'"`**
  This will filter for every product that starts with 'EXAMPLE_PHONE_'. You can only use the * sign at the end of the string.

Note

The order of the simple conditions does not matter, as they will be automatically grouped in the selection table.

In the background, in the ABAP selection table, two rows are related in the following way:

- Selections to the same field will have an OR relation.
- Selections to different fields will have an AND relation.

Because of this logic, using the operators AND and OR are not allowed.

In addition to these filtering techniques, there are also specific methods available for working with date fields in the system.

Filtering for `KEYFIGUREDATE` can be done by setting a value where it is filtering for the `KEYFIGUREDATE` in a **`yyyymmdd`** format.

Example

```
FilterForTheDataSource = "KEYFIGUREDATE BT '20160101' '20250101'"
```

5.6 Working with Extensions

Parameters for extensibility allow you to specify additional attribute mappings and filters that can be used to integrate data from external sources. You can further modify the way data is mapped to integrate data by using field extensions.

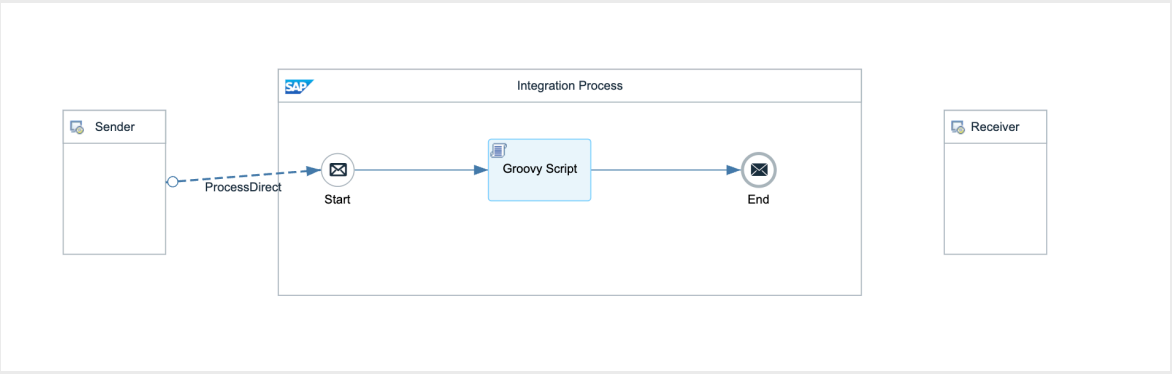
In general, the extensions are custom-defined integration flows that are called by a *Process Direct* call. In the `PostFetchExtensionIntegrationFlowAddress` and/or the `CustomMappingExtensionIntegrationFlowAddress` parameters, you must only add the flow ID of the custom extension integration.

5.6.1 Post Fetch Extension

Example

This example shows you how to manipulate data via a Groovy script.

A simple integration flow has to be defined, which is called by a *Process Direct* call. On the *Connection* tab, in the *Address* field, you enter the ID of your extension integration flow.



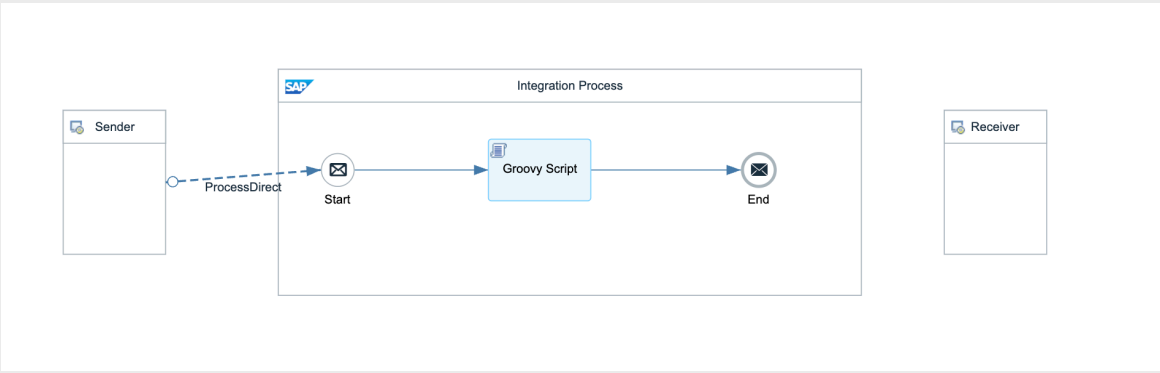
In the integration process, you need to define a Groovy script.

Example

This example shows you how to manipulate data via a Groovy script.

A simple integration flow has to be defined, which is called by a *Process Direct* call.

You must only enter the integration flow ID of the custom extension integration flow to the `CustomFetchExtensionIntegrationFlowAddress` parameter of the business iFlow.



In the integration process, you need to define a Groovy script.

For more information about the use cases of script, see SAP Cloud Integration - Script Use Cases.

For more information about the SCRIPT APIs, see https://help.sap.com/doc/a56f52e1a58e4e2bac7f7adbf45b2e26/Cloud/en-US/index.html.

The `Custom Mapping Extension` can also be implemented as a custom integration flow, called by a *Process Direct* call.

You must only enter the iflow ID of the custom mapping extension integration flow to the `CustomMappingExtensionIntegrationFlowAddress` parameter of the business iflow.

You can find an example *Message Mapping* among the *Resource* of the iflow. You can download this mapping and use it in a *Message Mapping* step in your custom integration flow. It serves as a basic template to help you understand the structure and logic required for your specific mapping needs. Apart from using a *Message Mapping* step, other ways are also viable, for example, using Groovy script to create the mapping logic that fits your requirements.

Note

The main difference between the `Post Fetch Extension` and the `Custom Mapping Extension` is that the `Post Fetch Extension` keeps the original mapping of the integration flow, while the `Custom Mapping Extension` you can still modify the data, but you must implement a mapping since the original business integration flow mapping is ignored.

Example

This example shows how to filter for `ExampleProduct` and modify the corresponding `Actuals Quantity` by adding **5** to it:

```
import com.sap.gateway.ip.core.customdev.util.Message
import groovy.json.JsonSlurper
import groovy.json.JsonBuilder

def Message processData(Message message) {
    // Parse the JSON payload
    def body = message.getBody(java.io.InputStream);
    def jsonPayload = new JsonSlurper().parseText(body.text)

    // Filter the records to include only those where PRDID is 'ExampleProduct' and add 5 to ACTUALSQUANTITY
    jsonPayload.ITEMS = jsonPayload.ITEMS.findAll { item ->
        if (item.PRDID == 'ExampleProduct') {
            item.ACTUALSQUANTITY = (item.ACTUALSQUANTITY as BigDecimal) + 5
            return true
        }
        return false
    }

    // Create a JSON builder to convert the map back to a JSON string
    def jsonOutput = new JsonBuilder(jsonPayload).toPrettyString()

    // Set the modified payload as the new message body
    message.setBody(jsonOutput)

    return message
}
```

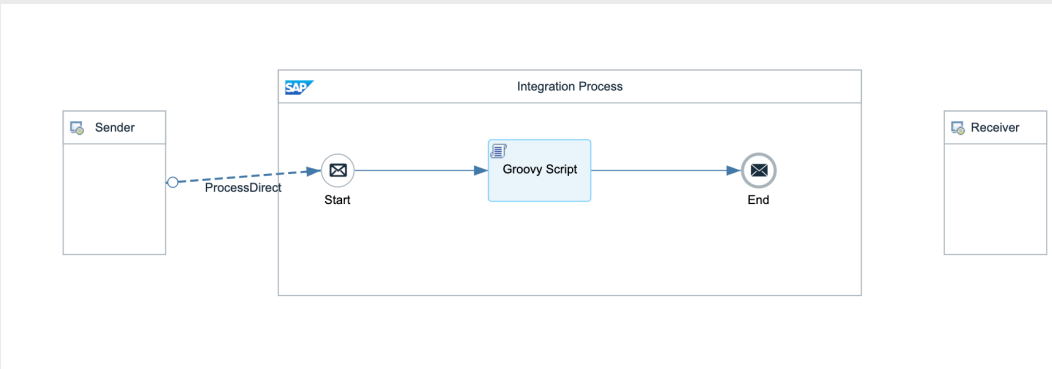
Creating the script is either possible in the SAP Integration Suite directly or by uploading a file created in an external editor. When using an external editor, make sure to include the `import com.sap.gateway.ip.core.customdev.util.Message` in the first row.

Example

This example shows you how to manipulate data via a Groovy script.

A simple integration flow has to be defined, which is called by a *Process Direct* call.

You must only enter the integration flow ID of the custom extension integration flow to the `CustomFetchExtensionIntegrationFlowAddress` parameter of the business iFlow.



In the integration process, you need to define a Groovy script.

For more information about the use cases of script, see SAP Cloud Integration - Script Use Cases.

For more information about the SCRIPT APIs, see https://help.sap.com/doc/a56f52e1a58e4e2bac7f7adbf45b2e26/Cloud/en-US/index.html.

5.6.2 Custom Mapping Extension

Similarly to the `Post Fetch Extension`, the `Custom Mapping Extension` can also be implemented as a custom integration flow, called by a *Process Direct* call.

You can find an example *Message Mapping* among the *Resource* of the *SAP IBP Add-On Read - Generic Process Flow*. You can download this mapping and use it in a Message Mapping step in your custom integration flow. It serves as a basic template to help you understand the structure and logic required for your specific mapping needs. Apart from using a *Message Mapping* step, other ways are also viable, for example, using Groovy script to create the mapping logic that fits your requirements.

> **Note**
>
> The main difference between the `Post Fetch Extension` and the `Custom Mapping Extension` is that the `Post Fetch Extension` keeps the original mapping of the integration flow, while the `Custom Mapping Extension` you can still modify the data, but you must implement a mapping since the original business integration flow mapping is ignored.

6 Troubleshooting and Error Handling

When working with the *SAP IBP Add-On Read - Generic Process Flow*, it is important to have a systematic approach to troubleshooting and error handling. This section provides guidance on identifying and resolving issues that may arise during the execution of the integration flow.

General Troubleshooting Steps

- Check the Integration Flow Statuses Across the Call Chain
  - Monitor the status of the integration flows in the SAP Cloud Integration *Monitor Message Processing* tool or in SAP IBP the *Data Integration Jobs* and *Application Logs* apps. In *Monitor Message Processing* use the `Correlation ID` to check every integration flow that was called during the process.
  - Look for any error messages or warnings that may indicate the cause of the failure.
- Examine Custom Headers and Attachments
  Inspect any custom headers or attachments that may contain additional error information or context about the failure.
- Verify Configuration:
  - Ensure that all required parameters are correctly set in the integration flow. Refer to the "Message Body Attributes" section for a list of mandatory parameters.
  - Confirm that the communication scenarios and connections between SAP IBP, SAP Cloud Integration, and the add-on are properly configured.
- Check Data Mapping and Filters
  - Validate the field mappings and filter criteria specified in the integration flow. Incorrect mappings or filters can lead to data processing errors.
  - Ensure that the `FieldMapping`, `SourceFieldsfromAddon`, and `TargetFieldsInSAPIBP` parameters are correctly defined.
- Review Extensions
  If using custom extensions, verify that the *Post Fetch Extension* and *Custom Mapping Extension* integration flows are correctly implemented and address the intended logic.

Specific Error Scenarios

- **Data Validation Errors**
  - If data validation fails during post processing in SAP IBP, review the error messages for specific validation issues.
  - Ensure that the data conforms to the expected format and business rules defined in SAP IBP.
- **Connection Issues**
  - If the integration flow cannot connect to SAP IBP or the add-on, check the network connectivity and authentication settings.

    - Verify that the SAP Cloud Connector is correctly configured and that the destinations are reachable.
- **Batch Processing Errors**
    - If batch processing fails, ensure that the batch parameters such as `BatchCommand`, `BatchName`, and `BatchKey` are correctly specified.
    - Check that the batch items contain valid data and that the batch size does not exceed system limits.
    - For SAP IBP post processing errors you can check the *Error Details*, *Custom Headers* and *Attachments* of the SAP IBP Write - Process Posted Data flow to

By following these troubleshooting steps and recommendations, you can effectively manage and resolve issues in the *SAP IBP Add-On Read - Generic Process Flow*, ensuring smooth and efficient data integration.

7 Recommendations

When integrating data with SAP IBP, 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, 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**
  Constantly 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.

www.sap.com/contactsap

© 2026 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.