Enterprise Talk

Planon Software Suite

Version: L107

About this Document

Intended Audience

This document is intended for *Planon Software Suite* users.

Contacting us

If you have any comments or questions regarding this document, please send them to: support@planonsoftware.com.

Document Conventions

Bold
Names of menus, options, tabs, fields and buttons are displayed in bold type.

Italic text
Application names are displayed in italics.

CAPITALS
Names of keys are displayed in upper case.

Special symbols

ℹ	Text preceded by this symbol references additional information or a tip.
❗	Text preceded by this symbol is intended to alert users about consequences if they carry out a particular action in Planon.

Table of Contents

About Enterprise Talk

The **Enterprise Talk** TSI enables users to import data from another application or data source into Planon ProCenter and to export data from Planon ProCenter to an XML/CSV.

Users can be authorized for using **Enterprise Talk**. The data imported into or exported from Planon ProCenter , however, is not subject to authorization.

Enterprise Talk is designed to import and retain all data until it stops. In other words, if the process is interrupted after importing 999 records, these 999 records will remain in the database.

Enterprise Talk can be used to achieve the following objectives:

- To support a Planon ProCenter implementation process by importing data from legacy applications into Planon ProCenter .
- To interface Planon ProCenter with other applications by sharing information. For example, by interfacing a Financials Management system with Planon ProCenter , you could import invoice amounts as actual costs into Planon ProCenter .
- Enterprise Talk is not meant to transfer configuration from test to production. Configuration information (FieldDefiner, TSIs, layouts, filters, alerts, reports, etc.) are meant to be transferred with Configuration Transfer.

For Business Intelligence tooling it is possible to export configuration and authorization information.

- To interface with mobile devices for survey management. Survey data is exported from Planon ProCenter and is imported into mobile devices ready for conducting maintenance surveys. When done, the survey data can be imported into Planon ProCenter .

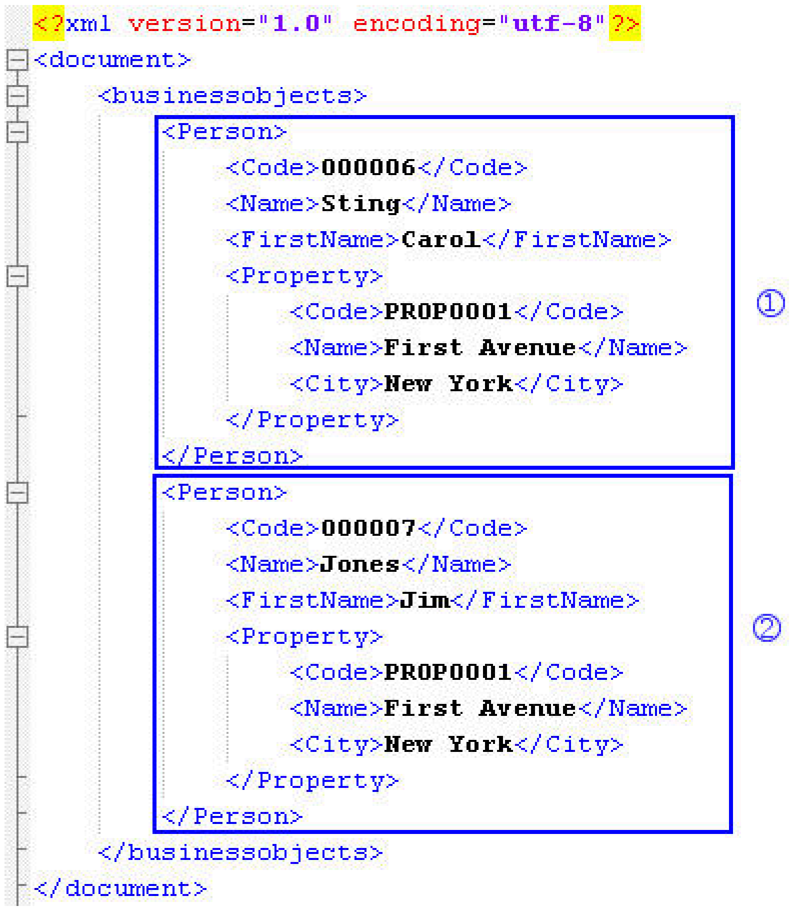
Enterprise Talk cannot be used to delete data from Planon.

SDI Configuration

Similar to **Enterprise Talk**, you can use **SDI Configuration** to import data in Planon. **SDI Configuration**, however, only works with data onboarding definitions, which are not shown in **Enterprise Talk**. For more information about **SDI Configuration**, see SDI Configuration.

Enterprise Talk - Architecture

- Importing/exporting files is handled by the application server, not by the client. Consequently, Planon ProCenter files will upload files from / download files to a location on the application server.
- Processing of XML elements is handled per main business object.

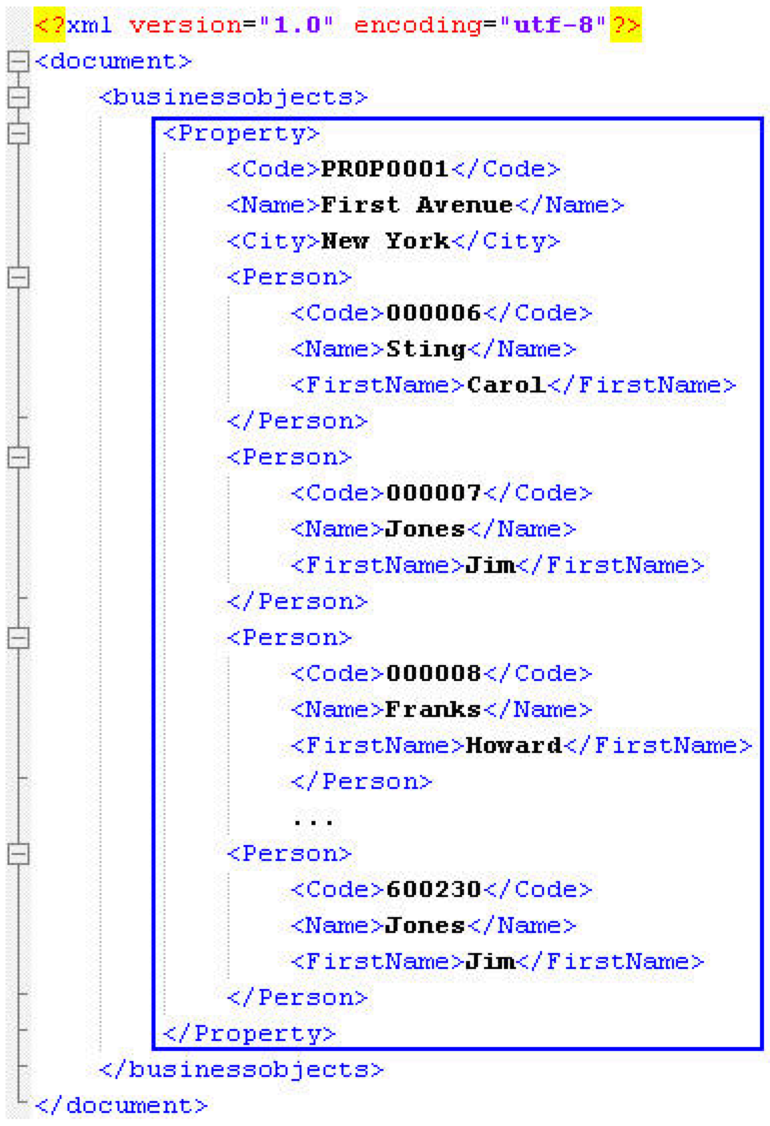


```
<?xml version="1.0" encoding="utf-8"?>
<document>
    <businessobjects>
        <Person>
            <Code>000006</Code>
            <Name>Sting</Name>
            <FirstName>Carol</FirstName>
            <Property>
                <Code>PROP0001</Code>
                <Name>First Avenue</Name>
                <City>New York</City>
            </Property>
        </Person>
        <Person>
            <Code>000007</Code>
            <Name>Jones</Name>
            <FirstName>Jim</FirstName>
            <Property>
                <Code>PROP0001</Code>
                <Name>First Avenue</Name>
                <City>New York</City>
            </Property>
        </Person>
    </businessobjects>
</document>
```

XML sample

The import will first read element **1** and then process element **2**.

Defining the main business object for an import file can affect the performance. If the main business object contains a limited amount of data, **Enterprise Talk** can quickly process this data. If it contains a whole range of data, this data will be stored in the cache till it is processed and this will affect performance.
For example, if Property is the main business object the XML contains a large data, and hence, the performance can suffer.



```
<?xml version="1.0" encoding="utf-8"?>
<document>
    <businessobjects>
        <Property>
            <Code>PROP0001</Code>
            <Name>First Avenue</Name>
            <City>New York</City>
            <Person>
                <Code>000006</Code>
                <Name>Sting</Name>
                <FirstName>Carol</FirstName>
            </Person>
            <Person>
                <Code>000007</Code>
                <Name>Jones</Name>
                <FirstName>Jim</FirstName>
            </Person>
            <Person>
                <Code>000008</Code>
                <Name>Franks</Name>
                <FirstName>Howard</FirstName>
                </Person>
                ...
            <Person>
                <Code>600230</Code>
                <Name>Jones</Name>
                <FirstName>Jim</FirstName>
            </Person>
        </Property>
    </businessobjects>
</document>
```

Main business object

Restrictions for importing/exporting data

For importing/exporting data, the following restrictions apply:

- The date format is fixed: YYYY-MM-DDTHH:MM:SS (XML standard).
- The decimal separator is a period (.) (XML standard).
- You cannot import business objects that are considered part of configuration, these must be imported with Configuration Transfer.
- Import/export does not work for non-database fields.

- The CSV export worker cannot handle nested data. Hence, first export to plain XML and transform it later to a CSV through XSLT.
- Account: Password only encrypted.
- MaintenanceActivityDefinitionTimeSchedule: If you import MADs and time schedules together, the following settings are ignored: "**If no records exists**", "**If records occurs once**" and "**If records occurs twice or more**". The default behavior will be that all the time schedules available in the import XML will be the end result in the database. All existing time schedules available before import will be deleted.
- You cannot directly **update** business objects that are subject to **audit logging** via Enterprise Talk (an error message will be displayed). Instead, you should first **delete** and then **insert** the business object. When this happens, a record of this is action correctly filed under the audit log.

  This restriction applies to the following M-to-N links of **User groups**:

  - Product definitions
  - Solution licenses
  - Users

Note that for the import definitions for these M-to-N relationships, both the *M* and *N* need to be part of search key.

Enterprise Talk is property-set-dependent. Planon ProCenter checks the property set of the logged in user during import-export. If you want to import/export a definition from/ to a different property set, you must change the property set by logging in on a different property set URL in the Web Client.

For specific restrictions surrounding business objects, see Business objects

Planon ProCenter Import policy

When importing data from an external source, it is possible that the data does not match the Planon ProCenter data specifications. To be able to correctly process data, the following import policies apply:

- If the import data exceeds the field length restrictions in Planon ProCenter , the data is truncated and a message is logged.
- If the import data is a string and has trailing spaces, these will be trimmed.
- When importing file paths, Planon ProCenter does not verify whether the path is valid.
- If a field cannot be imported, the composite business object will be skipped and the process will proceed with the subsequent business object and an error will be logged. This rule also applies to reference fields.

- When trying to insert values into a read-only field, the business object is skipped and an error is logged. When a read-only field is used as part of the search key, it will be used as a search criterion only, its value will not be updated, and no error is logged.

As an exception, there is a limited number of read-only fields that can be imported, see System fields/read-only fields that can be imported.

- If the XML lacks data to be imported into mandatory fields, the business object is skipped and an error is logged.
- For the field as sub, first, the respective (sub-) business object's import policy is applied, and the field is filled. Then, when none, two, or more results are encountered in the database, the value cannot be entered into the reference field, the business object is skipped and an error is logged.
- For the field as super, when none, two or more results are encountered in the database, the value cannot be entered into the reference field, the business object is skipped and an error is logged.
- In the XML, if the value for a field is empty, the empty value will replace the value in the database when importing the XML file.

In addition to these default field import policies, the user can also define import policies on **Business object definitions** level. See also Adding a business object definition (import).

Encoding

When importing diacritic characters in a CSV / Excel file using Enterprise Talk, the file encoding must be UTF-8 (with or without BOM). Otherwise, the diacritic characters will not be displayed correctly. The encoding can be converted in any text editor.

Data transformation

Some data transformation is currently possible:

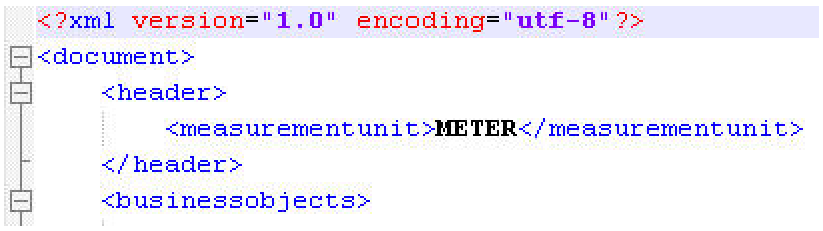
- Unit of measurement transformation on import
- Data file transformation

Unit of measurement transformation

Planon ProCenter supports the data conversion from meters to feet and vice versa. To be able to convert the measurement unit data during import, the XML header must specify the source measurement unit.

Example

If the database's unit of measurement is FEET, the import file's header should specify METER as follows:



```
<?xml version="1.0" encoding="utf-8"?>
<document>
    <header>
        <measurementunit>METER</measurementunit>
    </header>
    <businessobjects>
```

Consequently, when importing the XML, the measurement unit values are converted to feet. For a description when unit of measurement data conversion occurs, refer to Unit of measurement transformation.

Data file transformation

By using the concept of workers it is possible to transform data and import that data into Planon ProCenter or to transform Planon ProCenter data in such a way that it may be imported into another system.

Transforming data

Business objects

In general, all configurable business objects (those visible in Field definer) can be exported and imported with Enterprise Talk. However, there are some exceptions.

The following non-configurable business objects cannot be imported and exported:

- **File locations attachments** business object (AttachmentFilePath).

The following non-configurable business objects can be exported (not imported):

- Authorization business objects
- Field definer configuration business objects
- TSI configuration business objects

The following configurable business objects cannot be imported:

- Business objects that you are not allowed to create, such as Base orders.

The following business objects still have issues being imported:

- Floors (inserting new floors on a specific reference date with Enterprise Talk reference date is not yet possible)
- Asset groups (InventoryItemGroup) component, the field **Asset ID** (InventoryItemRef) cannot be set with Enterprise Talk yet

- Standard workflow objects (BaseStandardFlowObject)
- Standard sequence flow (StandardSequenceFlow)

Fields

This section provides an overview of the field types that can be used by Enterprise Talk and those that cannot be used.

Supported fields

- String (also extended string)
  - URLFieldDefinition
  - File reference fields such as images, documents and AutoCAD files.

For file fields, the file location folder is not exported:

Example
- image file location: \\company\FileLocations\Images
- image field: buildingX.png
- Planon ProCenter shows: \\company\FileLocations\Images\buildingX.png
- Enterprise Talk exports: buildingX.png

- StringReferenceField (pick lists)
  - For example picklists, city reference fields.
  - You cannot use this field as sub or super, you can only export/import the lookup.
  - You should include the complete BO as main BO:
    - In the import: if to be created/updated during import.
    - In the export: to use additional info next to the lookup outside Planon.
- Boolean (accepted values are: true, false, 1 (for true) and 0 (for false))
- Integer
- Big decimal

The following convention applies to big decimal fields: • The decimal separator is always a period (.)
- A thousand mark is not supported.
- There is no support for scientific notation.

- Monetary fields: same as big decimal.
- Volume: same as big decimal, should take into account measurement unit, if defined.
- Area: same as big decimal, should take into account measurement unit, if defined.
- Length: same as big decimal, should take into account measurement unit, if defined.
- Tariff per unit area (M2Tariff): same as big decimal, should take into account measurement unit, if defined.

- Integer reference field: lookup field (import and export) or as sub BO included.
- Status fields: it is possible to implement a status transition via import. Add 'Status' as 'sub' and 'System name' and 'Business object' fields as a part of the search key.

  For export, use a specific filter for 'Status' field instead of 'System status'. Example, for Budget, use (Status = BC10, Initial)) instead of (System status = 1, Initial).
- Date time fields:
  - Date neutral (CCYY-MM-DD)
  - Date time neutral (CCYY-MM-DDThh:mm:ss)
  - Time neutral (hh:mm:ss)
  - Date time property (MTZ field, currently the same format as for neutral)
  - Date time transaction (MTZ field, currently the same format as for neutral)
- Period
- MonthsOfYear: In DB x.x.x.x.x.x.x.x.x.x.x.x.x.x.x, this will be translated to:
  - January, February, March etc.
    - seperated with spaces
    - not duplicated
    - not case sensitive
    - no more than twelve months
    - any order possible
  - For export and import
- MultiSelectDateField (Dates in format yyyymmdd separated by a space)
- AddressType: import expects a value as it exists in the database. Each address type has a character: CILOUPQRST. The value in the database is a string containing the characters for the selected address types and a space for non-selected address types. For example: for address with address type 2, 5 and 9 the value is “I U S”
- PersonType: import expects a field value as exists in the database. Each person type has a number (1-10). The value in the database is a string containing the number for selected person types and a space for the non-selected person types. For example, for a person with person type 2, 5 and 9 the value will be “2 5 9”.

System fields/read-only fields that can be imported

It is possible to import specific system fields/read-only fields for a restricted number of business objects. The following table lists the business objects and fields for which this is possible:

Business object	System name of field
CADFieldMapping	DefaultValueInXML
BaseBudget	BudgetCategoryRef
StandardOrder	BusinessObjectDefinitionRef
StandardOrder	RefBODefinitionUserDefined
WorkSpace	EndDate
BaseQuestion	DefaultValueXml
For the user-translated language BOs	P5LanguageDefinitionRef
For the user-translated language BOs, e.g. LangBaseAsset	The reference to the BO itself, e.g. if assets is user-translated, then the BO LangBaseAsset will have a BaseAssetRef

Unsupported fields

- Typically, system fields/read-only fields are not supported to be imported with Enterprise Talk. However, as discussed previously, there is a list of exceptions.
- The filter (SearchCriteria) field is not supported.
- CSS field.
- Blob field.

Enterprise Talk - Concepts

The following concepts are explained in this section:

- Bucket
- Chunk
- Composite business object
- Data source
- Document source
- Enterprise Talk reference date
- ETL
- Field as super/sub
- Import/export Definition
- Inbound marking
- Main business object
- Nested data
- Part of search key
- POJO
- S3
- Worker
- Worker bundle
- Workers chain
- XSLT

Bucket

A bucket is a container for objects. Amazon S3 stores data as objects within buckets. An object consists of a file and optionally any metadata that describes that file.

For each bucket, you can control access to it (who can create, delete, and list objects in the bucket), view access logs for it and its objects, and choose the geographical region where Amazon S3 will store the bucket and its contents.

For more information on creating and configuring an S3 bucket, see http://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-configure-bucket.html.

Chunk

A chunk is a set of elements comprising a subset of an import- or export document. A chunk's number of items is determined by the **Fetch size**.

A chunk can be:

- a number of XML elements in case of a single document
- a number of files in case of a single document and a fetch file worker

If a definition contains two documents and the fetch size is set to 100, it will process chunks of 100 elements until the first document is done and will then move on to process the elements of the second document until it is done too.

Composite business object

A business object that contains data of another business object.

Data about a concept may be split over multiple business objects. Conceptually, the data is related; technically, the business objects are split.

The rules of logic determine that for a composite business object, the related business objects cannot exist without the main business object.

Examples

The Orders business object consists of data from the Order lines business object. The Order lines business object cannot exist without the Orders business object.

Importing into multiple business objects – Field as sub

Importing into multiple business objects – Field as super

Nested data

Data source

The data file for creating import/export definition.

You specify the data source while adding import/export definitions in the **Documents** level.

While exporting data, you can generate uniquely identified files (xml, csv) per business object in the file path specified in the **Data source** field. The files are saved with a date/time stamp so that when different filter criteria are used to export data, the files are easily identified.

Document source

Describes the document source to be imported/exported.

An import/export document provides the data source to be imported from or exported to for running an import/export definition. For example, xml, csv, excel, databases.

Enterprise Talk reference date

A date filter that allows you to access time-dependent data. By setting a reference date you can export/import elements that are valid on this date. The following applies to an import:

- The reference date is used a start date.
- The reference date specified in the import definition is leading over the date specified in the XML.

Business objects that support the reference date concept have a start date and an end date. The reference date is also used for life cycle business objects.

ETL

Acronym for extract, transform, and load.

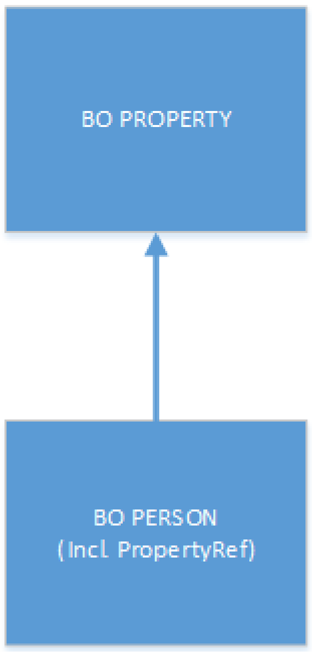
A generic term that is used for the concept of extracting data from one source, transforming it to fit operational needs, and loading it into the target. **Planon Enterprise Talk** is Planon ProCenter built-in ETL tool.

Field as super/sub

- When exporting business objects you can determine the hierarchical structure in the XML output by specifying the **Field as super** or **Field as sub**.
- When importing hierarchical business objects, you must define the structure correctly in the **Field as super** or **Field as sub** definition.

Example

This example outlines how to export a person and the property to which this person is assigned.



If the exported XML should look as follows:

```
<businessobjects>
  <Person>
    <Code>127</Code>
    <LastName>Alkin</LastName>
    <FirstName>Matt</FirstName>
    <Property>
      <Code>14</Code>
      <Name>Columbus Square</Name>
      <City>London</City>
    </Property>
  </Person>
</businessobjects>
```

- Then in Enterprise Talk go to the business object that contains the reference field: BO **Person**.
- Define the **PropertyRef field** as a **Sub**, because in the XML hierarchy, the property is at a lower level than the person (property is *nested* inside BO **Person**).

If the exported XML should look as follows:

```
<businessobjects>
  <Property>
    <Code>14</Code>
    <Name>Columbus Square</Name>
    <City>London</City>
    <Person>
      <Code>127</Code>
      <LastName>Alkin</LastName>
      <FirstName>Matt</FirstName>
    </Person>
  </Property>
</businessobjects>
```

- Then in Enterprise Talk go to the business object that contains the reference field: BO **Person**.
- Define the **PropertyRef** field as a **Super**, because in the XML hierarchy, the property is at a higher level than the person (person is *nested* inside BO **Property**).

The same principle applies to importing a business object.

Import/export Definition

A placeholder for configuring the import/export process.

In Planon ProCenter , this is the first level in the **Planon Enterprise Talk** process. At this level, you define the Import/Export definition and run it from the action menu.

Inbound marking

Identifying business objects during import.

A mechanism for marking business objects that have been imported earlier, but that are no longer available in the new import. In these business objects you can set a field to a specified value.

When importing business objects, the **Marking current state** is used to see which business objects should be marked.

When **Enterprise Talk** starts an import for a specific business object and marking is switched on, the **Marking current state** is set to 1 for all business objects having value 2.

When a business object is imported (inserted or updated), the value of the field **Marking current state** is set to 2.

When all business objects have been imported, **Enterprise Talk** will look for business objects that still have value 1. These are the business object that were no longer in the import. **Enterprise Talk** will put the value specified in **Mark value** in the field **Mark field** and the **Marking current state** is set to 2 again.

Main business object

Identification of the main element in an import/export XML file.

If an XML file contains data belonging to two or more business objects, you can import data for these two or more business objects. The main business object identifies the first element for processing. In an export file, the main business object will be the main node in the XML file.

Enterprise Talk - Architecture

Nested data

Data in a hierarchical structure in an XML file.

If an XML file contains data belonging to two or more business objects, you can import data for these two or more business objects.

Examples

An XML file containing orders and their related order lines.

Importing into multiple business objects – Field as sub

Importing into multiple business objects – Field as super

Composite business object

Part of search key

A method to check whether a correct item is found when looking up the business object in a list. You can specify multiple fields to identify a business object.

POJO

An object that consists of data from an XML element.

When processing XML files, the XML elements in these files are converted to objects that can easily be manipulated. These objects are called POJOs.

Creating customized workers

S3

Amazon Simple Storage Service (Amazon S3) is the cloud storage solution of Amazon Web Services (AWS).

S3 has a web services interface that you can use to store and retrieve any amount of data in buckets, at any time, from anywhere on the web.

For more information on Amazon S3, see http://aws.amazon.com/s3/.

Worker

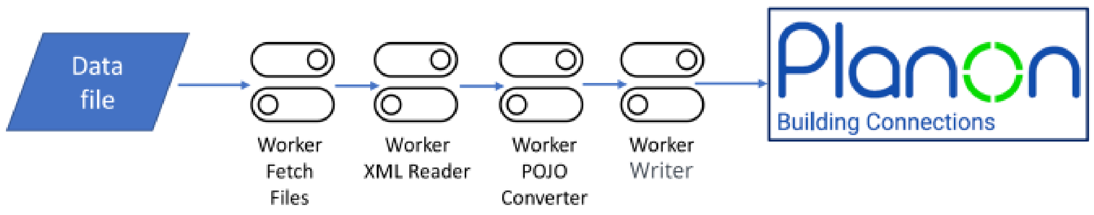
Software code that performs a specific operation on data.

The data to be imported into Planon ProCenter may come from various sources and formats. To be able to import and export data of different formats, Planon ProCenter comes with a number of workers that perform a specific operation on the data file.

Using standard workers data can be imported from and exported to XML and CSV file formats.

Custom workers can be written to import from and export to various source/destinations such as, flat file, Excel and database table.

Planon ProCenter features the following default workers for import:



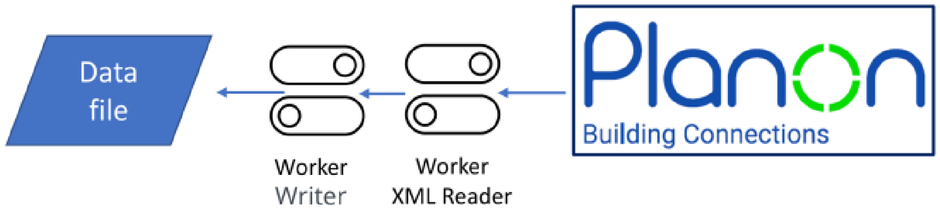
Schematic overview - import

- FetchFilesWorker

The worker fetches the file(s) for import. This worker may be used only if you want to read multiple files at one go during the import.

- XMLReaderWorker
- This worker reads the XML file.
- XMLToPOJOConvertor
  This worker converts the XML elements to an object that is understood by workers of Planon ProCenter .
- PlanonWriterWorker
  This worker writes data into Planon ProCenter .

  Planon ProCenter features two default workers for export:



Schematic overview - export

- **PlanonReaderWorker**
  This worker reads data from Planon ProCenter .
- **FileXMLWriterWorker**
  This worker writes data to an XML file.

  As these workers carry out specific operations, the sequence in which they carry out these operations is important. The process of transforming and importing data succeeds only if the proper sequence is adhered to.

  In addition, Planon ProCenter features the following workers that may be used in the process to perform further data transformation:

- **XSLT transformation**

  FileXSLTTransformationWorker
  The XMLReaderWorker can read XML files in a specified format only. The format depends on the configuration of the business objects done in the Enterprise Talk. Sometimes, the data to be imported from an XML file need not be in the specified format. This worker reads a file and transforms it into the 'Reader' understandable format, so that the 'Reader' can process it further.

For import, FileXSLTTransformationWorker can be used to complete the whole transformation process within a single document.
For export, you should create a single definition with two documents: one to write data from Planon and convert it to XML, and a second, using only FileXSLTTransformationWorker, to convert the XML output from the first document to a transformed XML.
See also FileXSLTTransformationWorker and Data transformation example (export)

XMLChunkXSLTTransformationWorker

This worker reads a main business object from the ‘Reader’ and passes it on to the POJO reader worker or vice versa.

- **CSV transformation**

  CSVReaderWorker
  This worker reads the CSV file and converts the CSV into XML elements.

  CSVWriterWorker
  This worker converts the XML elements and writes them into a CSV file.

Customers may create their own workers that interact with the Planon ProCenter workers. For more information on creating your own workers, see Creating customized workers.

Transforming data

Creating customized workers

Worker bundle

A set of workers created to perform a specific purpose.

Because data sources can vary in content and format, specific data manipulation may be required to transform data into the desired format. A worker bundle provides a specific set of workers designed to interact and to bring about the desired output.

The workers specified as part of a definition need to be sequenced in order to chain and process them in the right order.

Creating customized workers

Workers chain

Each worker works with a main business object's data at a time. After processing the data, the transformed data is passed on to the next worker. Once the last worker in the chain has completed processing the data, the workers are checked to see if all data has been processed, in reverse order starting from the last worker. If any worker has not completed processing the data, the worker is executed again and after that, the next worker in the chain is called. This process continues until the first worker reports that it has completed processing the data.

Creating customized workers

XSLT

Acronym for Extensible Stylesheet Language Transformation.

An XML-based language used for the transformation of XML documents into other XML documents. The original document is not changed; rather, a new document is created based on the content of an existing one.

Examples - data transformation

Working with Enterprise Talk

The following sections describe how to set up and work with Enterprise Talk.

- Configuring
- Features
- Importing
- Exporting

Configuring

The following sections describe how to configure Enterprise Talk:

- Adding a definition
- Exporting a definition
- Importing a definition
- Adding a document
- Copying a document
- Adding a business object definition (import)
- Adding a business object definition (export)
- Adding field definitions
- Adding fields as sub
- Adding fields as super

Adding a definition

To enable the configuration of the import/export process in Planon ProCenter :

Procedure

1. Go to Definitions.
2. To add an import/export definition, enter a Code, Description and, optionally, a Comment.
3. Click Save.

For more information about the fields at this level, see Definition fields.

After completing the definition configuration on the other selection levels, on the action panel, click **Run** to run the definition.

Exporting a definition

To enable to share a data import/export definition across Planon ProCenter environments:

Procedure

1. Go to Definitions.
2. Select the definition that you want export.

Use SHIFT+click or CTRL+click to select and include multiple definitions.

3. On the action panel, click Export definition.

A zip file including the definition and the export.info file appears in your browser's download location.

Importing a definition

To enable to reuse an existing data import/export definition.

Procedure

1. Go to Definitions.
2. On the action panel, click Import definition. The upload file dialog box appears.
3. Click Open and select an Enterprise Talk zip file.

The zip is unzipped the export.info is read and the XML files listed in export.info are read (see Exporting a definition).

4. When the Enterprise Talk definition is overwritten (not added) a confirmation dialog box appears:

Talk definition <ID value> will be overwritten, continue?

5. If you click Yes: the Enterprise Talk definition is imported.

The Importing pop-up appears, displaying a progress bar. When the import(s) is/are successful, a Successful pop-up appears. When the import encountered errors, an Errors popup is displayed.

Adding a document

To define import/export document in order to be able to run an import/export definition.

Procedure

1. Go to Documents.
2. To add an import/export document, fill out the fields.
3. For a description of these fields, refer to Document fields.
4. Click Save.

Adding a document

Adding a business object definition

Adding a business object definition (export)

Adding default values to fields during import

Using chunks to prevent a transaction time-out

Import-/export documents are processed in chunks. You can define the size of these chunks by specifying the **Fetch size**. The chunks will be processed as separate transactions within the same run. This feature applies to both manual and scheduled runs.

For more information about the **Fetch Size** field, refer to Document fields.

Copying a document

In order to save time in configuring a definition, you can "(deep) copy" a document to an existing definition and reuse it.

1. Export the Enterprise Talk definition whose document you want to copy.

   The definition and all its details will be exported to the location you select.
2. Temporarily change the Code field of the definition that you have just exported (e.g. add the suffix "_origin").
3. Import the definition you have just exported (Step 1).

   The new definition will appear in the elements list.
4. Select the new definition and go to the Documents level and select the document you want to copy.
5. In the Import-export definition field, select the definition to which you want to link the document.
6. Go back to the Definitions level and delete the definition you have just imported (Step 3).
7. Rename the original definition's Code field to its original name (remove the suffix "_origin").

You have completed "copying" a document to another definition and you can now continue to fine-tune its configuration.

Adding a business object definition (import)

To specify the business objects into which you want to import data.

Procedure

1. Go to Business object definitions > Business object definitions.
2. To add an import business object definition, on the action menu, click Add import business object definitions and complete the fields. For a description of these fields, refer to Business object definition fields.
3. Click Save.

You can add field definitions at this level. On the action menu, under **Links**, click **Business object definition fields**. This allows you to add multiple fields in one go. Alternatively, you can add field definitions at the **Definition details** level where you can also specify fields as sub or super.

Adding a business object definition (export)

To specify the business objects whose data you want to export.

Procedure

1. In Enterprise Talk > Enterprise Talk > Business object definitions.
2. To add an export business object definition, on the action menu, click Add export business object definitions and complete the fields.
3. For a description of these fields, refer to Export business object definition fields.
4. Click Save.

You can add field definitions at this level. On the action menu, under **Links**, click **Business object definition fields**. This allows you to add multiple fields in one go. Alternatively, you can add field definitions at the **Definition details** level where you can also specify fields as sub or super.

Adding a document

Adding field definitions

Using the Enterprise Talk root folder

In Enterprise Talk you can point to your data source file by using absolute path names (such as C:\My documents\Person.xml). However, in some situations, using absolute path names does not work or limit the functionality (disable the use of WebDAV folders). This section explains how you can use relative path names to refer to your data source.

If the **Enterprise Talk root** setting (**System Settings**) is set, this folder is used to locate the data source.

If the **Enterprise Talk root** setting is empty, the location should be relative to the Server \tanuki\appserver\bin folder

The following table explains how this works. The value **Data source** field is: ..\..\Talk \person.xml

Enterprise Talk root	Data source location
<empty>	Server\tanuki\Talk\person.xml
\\ComputerName\SharedFolder\Talk	\\ComputerName\Talk\person.xml

When using an absolute path name, you cannot use webDAV locations. However, it is possible to specify a webDAV folder in the **Enterprise Talk root** setting.

For more information about this setting, see *System Settings*.

When going from fixed, or relative file locations in Talk to an absolute **Enterprise Talk root** folder (for example a WebDAV location):

- Set the absolute location in the **Enterprise Talk root** folder in **File locations** (**System Settings**).
- Then change all folders and file locations in Talk to relative folders.

Adding field definitions

To specify the fields that should be exported / imported for a specific business object.

You can also add field definitions at the **Business object definitions** level. On the action menu, under **Links**, click **Business object definition fields**. This allows you to add multiple fields in one go, although it does not allow you to specify fields as super nor sub.
The filter (SearchCriteria) field is not supported.

Procedure

1. Go to Definition details > Detail definitions.
2. To add field definitions, click Add Field definitions on the action menu, and complete the fields. For a description of these fields, refer to Field definitions & default values to fields.
3. Click Save.

Add a definition for each field for which you want to import or export values.

The order in which fields are exported is on syscode. If you want to change the order of the imported fields, in the XML definition, first remove them and then add them again in the correct order.

Adding a business object definition (export)

Adding default values to fields during import

Adding default values to fields during import

Sometimes, during an import, the import file may not contain information about fields that are mandatory or fields where you want to have data in. When an import file does not contain information about these fields, you can set default values on such fields. These values are then received as part of the imported data.

For example, if you are importing **Person** BO, and the **City** field does not contain a value in the import file, you can select a default value for the city.

Typically the default value from Field definer is used, unless overruled by the default as specified in the Enterprise Talk field definition.

To add import default value to a field

Procedure

1. Go to Business object definitions > Definition details.
2. Select the Import definition detail for which you want to add a default value and complete the fields in the data section.

   For a description of these fields, refer to Field definitions & default values to fields.
3. Click Save. The default values are defined on the import BO definition.

Adding a document

Adding field definitions

Adding field as sub

To specify the fields that should be exported / imported for a specific business object. By specifying fields as sub, you link two related business objects via a reference field (see also Field as super/sub).

On the **Definition details** level, you can add fields as sub in relation to a business object definition.

For example, you can add the Property business object as sub to the Person business object. By doing so, you are importing a list of persons and the properties to which they are linked.

In the XML file, the fields in the Property node are in a lower hierarchy in relation to the Person node. For an XML example, see Import file.

Procedure

1. Go to Definition details.
2. To add field as sub, click Add field as sub on the action menu, and complete the fields. For a description of these fields, refer to Field as sub.
3. Click Save.

Add a definition for each field for which you want to import or export values.

Importing into multiple business objects – Field as super

Importing into multiple business objects – Field as sub

Adding field as super

To specify the fields that should be exported / imported for a specific business object. By specifying fields as super, you link two related business objects via a reference field (see also Field as super/sub).

On the **Definition details** level, you can add fields as super in relation to a business object definition.
For example, you can add the Property business object as super to the Person business object. By doing so, you are importing a list of properties and, per property, the persons linked to the properties.

In the XML file, the fields in the property node are higher in the hierarchy in relation to the person node. For an XML example, see Import file.

Procedure

1. Go to Definition details.
2. To add field as super, click Add field as super on the action menu, and complete the fields. For a description of these fields, refer to field as super.
3. Click Save.

Add a definition for each field for which you want to import or export values.

Importing into multiple business objects – Field as super

Importing data into a single business object

Features

The following sections describe specific features in Enterprise Talk:

- Automatically running an import/export definition
    - Adding an action definition
- Transforming data
- Specifying the reference date
- Logging
    - Deleting logs
    - Automatic deletion of expired logs
- Updating an existing business object
- Transforming data
- Running a Talk definition.

Automatically running an import/export definition

Once an import/export definition is configured, you can run it manually or automatically.

For running a definition manually, select it on the **definitions** selection level, and on the action menu, click **Run**.

For running a definition automatically, additional configuration steps are required. This section describes how to add an action definition in order to automatically run a data import/export definition.

For more information on action definitions, see the Alerts documentation.

Adding an action definition

Procedure

1. Go to **Action definition**.
2. Click Add on the action panel.

a. Click the Select a value button in Alert condition, the Alert condition dialog box appears.

b. Click Add. Select the ImportexportDefinition business object from the list field. Enter a code for the alert condition. Specify a filter on the Code field to select the correct definition. Fill out any other details and click OK.

You return to the action definition layout.

3. Select an action from the list in the Action field. Select Run (BOMExecute).
4. Click OK.
5. Select a schedule for the action definition to be run. You can select Minutes/Hourly/Daily/Weekly/Monthly or from the Calendar.
6. Select the start/date time to indicate when the action definition should start.

   If, for example, you have a daily data interface with an external system, choose your start time at a time when there are few people active in the Planon system so as not to affect their performance.

7. Fill out any other details for the action definition.
8. Click Save.
9. Finally, to make the action definition available, click Active on the action panel.

You have completed configuring a scheduling job for your import/export definition. The job will run automatically according to the schedule configured for the definition.

Transforming data

Since data transformation works in two directions, the goal is twofold:

- To be able to import data from a file whose contents is not compatible with Planon ProCenter .
- To be able to export data from Planon ProCenter to a file.

Workers are a core concept in Planon ProCenter 's **Enterprise Talk**. Default workers to support import/export are available. These default workers are enabled upon adding a document on **Documents** level.

Import

If data in the data file to be imported is compatible with Planon ProCenter , no further data transformation is required, and you can use the default workers to import your data.

Most likely, the data in your file is not compatible with Planon ProCenter , and transformation is required to import the data. Planon ProCenter uses workers to perform data transformation. Here, you can add your own, customized workers.

Export

If you want to export data from Planon ProCenter and no further data transformation is required, you can use the default workers to export your data.

To transform the data exported from Planon ProCenter , you can add workers to perform data transformation. Here, you can add the required workers in the appropriate order.

To be able to add your customized worker here, it must first be added to the server\default \deploy\bundle\planon folder

1. To add a worker, on the action menu, click Add and complete the fields.
   For a description of these fields, refer to Transforming data.
2. Click Save.

   Data transformation example (import)

   Creating customized workers

Data file transformation

Worker

Specifying the reference date

There are various ways in which you can specify the reference date.

- By specifying a specific date on the **Documents** level by using the **Reference date** field's date picker.

  If you specify a specific date, the export will display data that is valid for the date specified. When importing, the date is used as a start date for the imported business objects (only if reference or life cycle aware).
- By using a date macro in the **Reference date** field on the **Documents** level.

  By using the &Date macro, you can export/import data that is valid on a specific date relative to the current date. This macro can be used in combination with date-related secondary arguments (for example: &Date &Y). Time-related macros are not supported.

The &Date macro is a new type of macro that is only valid for Enterprise Talk.

For more information on macros or secondary arguments, see *Fundamentals*.

- By not specifying a date in the definition.

  If you do not specify a date, your export will contain all data, including all life cycle entries. Similarly, for reference date aware business objects, all occurrences will be exported.

Logging

To provide information to the user on the status of the import or export.

On the **Business object definitions** step, select one, none, or all definitions and click the **Document log** step to view all respective log messages.

If, on **Documents** level, you select the **Log successful imports** field, the **Document logs** step will contain an import summary only. On the**Definition details** level, no further data will be shown.

Logging is done for:

- Errors
  : listing all errors encountered during the import.
- Warnings
  : listing all warnings encountered during the import.
- Information: providing a summary of the import

For more information on the fields for logging, see Log / Log details fields

Download logs

You can also download the logs as a PDF.

1. Go to Business object definitions > Import document logs and select the logs that you want to view and click **Download logs.**

   These logs will be combined in a single PDF, which you can download from or view in your browser.

The **Download logs** action only properly works for CSV and Excel imports and not so much for XML.

Additional debug logging

If you have access to your own test environment, you can add additional debug logging parameters that will log additional run information in your application server log. In the application server service configuration file, add the following line:

```
-Dnl.planon.hades.dataimportexport.track.Tracker
```

If you sometime need more detailed info (log each filled field), then also add the following line:

```
-Dnl.planon.hades.dataimportexport.track.Tracker.VerboseLogging
```

This additional logging will be logged after restarting the application server.

Download latest log

To get immediate feedback on the results of the latest run, you can download the latest log.

1. On **Definitions** level, select your import definition and click **Download latest log.**

   This will always download the log of latest action that is performed. The log itself is a formatted PDF, which you can view in your browser or by using a PDF viewer.

   Example

```
Summary of total import:

Document: Employee

Start date-time: 2018/03/01 10:57:59 GMT+5:30

End date-time: 2018/03/01 10:57:59 GMT+5:30


UsrEmployee:

Number of business objects processed: 4

Number of business objects skipped: 2

Number of business objects created: 0

Number of business objects updated: 2


FloorRef:

Number of business objects processed: 2

Number of business objects skipped: 2

Number of business objects created: 0

Number of business objects updated: 0

SpaceRef:

Number of business objects processed: 2

Number of business objects skipped: 2

Number of business objects created: 0
Number of business objects updated: 0


Issues found...

UsrEmployee:

Field Code is empty or contains invalid data.(PN_H00002)

for Row 1 to row200, Row250.


UsrSpace:

Field Code is empty or contains invalid data.(PN_H00002)

for Row 1 to row200, Row250.


Source Imported

-------------------- ---------

UsrEmployee new.xlsx Failed
```

The information in this log is as explicit as possible and is clubbed by row.

Download logs

In addition to viewing the logs in the application, you can also download the logs as a formatted PDF.

This feature is available for:

- Enterprise Talk / SDI Configuration > Business object definitions > Import document logs
- Data Onboarding > Logs
- CAD Import > Import logs

1. Go to the appropriate level (see earlier) and select the log(s) that you want to view.

If you select multiple logs, these will be combined in a single PDF.

2. Click Download logs. Your PDF will be downloaded and you can view it in your browser or by using a PDF viewer.

The **Download logs** action only properly works for CSV and Excel imports and not so much for XML.

Logging configuration

To apply formatting improvements to logging in **Enterprise Talk**, some configuration is required.

Formatting improvements that work out-of-the-box for **SDI Configuration** and for **Data Onboarding**, require some configuration for **Enterprise Talk.**

1. For the PlanonDataWriterWorker, in the Additional parameters box, add the following line:

```
consolidateLogs = true
```

2. Click OK to save your changes.

This will then work for the ExcelReaderWorker and for the CSVReaderWorker. By default, these two workers will then add an additional attribute to the main business object XML created from a single row named row_id.

For a custom worker, if the main business object has this attribute row_id=1 and consolidateLogs = true in PlanonDataWriterWorker, then this will also work.

Deleting logs

The logs created while importing/exporting are automatically deleted using a scheduled task. For more information, see SYSEVENTLOG_CLEANER (System Settings).

Uniquely identifying a business object

In order to update an already existing business object in Planon, Enterprise Talk must be able to uniquely identify it. This can be done by using the **Part of search key** field.

In Enterprise Talk, often the **Code** field is a unique field. When setting the **Part of search key** field to **Yes** for the Code field definition, an existing business object will be updated with the data from the import file when the code field from the business object matches the code value in the import file.

Sometimes, however, the **Code** or any other field does not uniquely identify a business object. This is for example the case for **BaseBudget** and **BOBaseBudgetCategory**.

The following example shows how to configure Enterprise Talk when using more than one field to uniquely identify a business object (using BaseBuget as an example):

1. On the Business object definitions level set the If record occurs once to 1, Create/update for the business object to update.
2. Determine which fields uniquely identity a business object.

Let's state that for BaseBudget the combination of Budget category, Financial year, Code and Description make the business object unique.

3. Go to Business object definitions > Definition details. If not available, add the field definitions for these fields:

   - Budget category

   - Financial year
   - Code
   - Description
4. In the field definitions of these fields, set the Part of search key field to Yes.

- The updating will only be done if the **If record occurs once** field on the **Business object definitions** level is set to ***1, Create/update***.
- If a field is marked as search key and a business object is found, this field will not be updated.
- Consequently, read-only or system fields can be used as search key.

For an example, see Importing a business object without a unique key field (Budget).

Running an Enterprise Talk definition

Click **Run** to run the Enterprise Talk definition manually. The process will be initiated as a background action. The progress of the Enterprise Talk definition process can been seen in the **Background action progress** bar in the **Background actions** TSI (or click **Refresh** in the elements panel). For more information about background actions, see Supporting data .

When you run an Enterprise Talk definition on multiple documents, note that the failure of a single document will stop the execution of the rest of the documents.

Canceling a process

After clicking **Run**, the import- or export is in progress, it's status will be *Running*. If it is taking too long or you have made a mistake, you can abort the process by clicking **Request to stop** on the action panel.

This status transition will become available once the **Run** button is clicked.

The process will be aborted and the status of the definition will change to *Stopped*.

- If not configured, you may need to add the status transition to the layout.
- This action should only be used for long-running definitions for which it takes longer to update progress.

When clicking **Request to stop** while the progress is being updated - an error will occur. To overcome this issue, you can increase the fetch size (on **Documents** level) and change it from 1000 to 2000 so that it takes longer to report progress updates and this process does not interfere with the **Request to stop**.

Importing

The following sections describe the import features of Enterprise Talk:

- Reading multiple files
- Importing/exporting linked business objects (m-to-n relation/multiple select free field links)
- Importing/exporting maintenance activity definition schedules
- Importing on a reference date
- Importing a contract including life cycles
- Importing different UDBO types in a single run
  - Updating different UDBO types in a single run
- Archiving/Dearchiving a business object

Reading multiple files

This feature allows you to read data from multiple files at the same time via a single import definition. This way you can avoid creating separate definitions for running each file individually. File types such as xml, csv, xls, xlsx are supported.

Reading multiple files can be done by executing below two steps:

1. *Select a wild card character * or ? to be used in the file names.*

   The '*' can be used for multiple characters and the '?' for a single character.
   For example,
   ../PlanonData\EnterpriseTalk\Person*.xml
   ../PlanonData\EnterpriseTalk\Person123?.xml
   When there are a lot of files matching a part of the file name mentioned in the **Data Source** field, the oldest file is read first and the same process continues.
   For example,

FileName	Modified Date Time
Person22.xml	22-07-2012 12:00:02:20
Person12.xml	22-07-2012 12:00:02:10
Person11.xml	22-07-2012 12:00:02:05

   Here, Person11.xml file will be read first.

2. *Use FetchFilesWorker, to read the (multiple) files.*

For more information, refer to FetchFilesWorker.

For more information on the Data source field during import/export definition, refer to FetchFilesWorker.

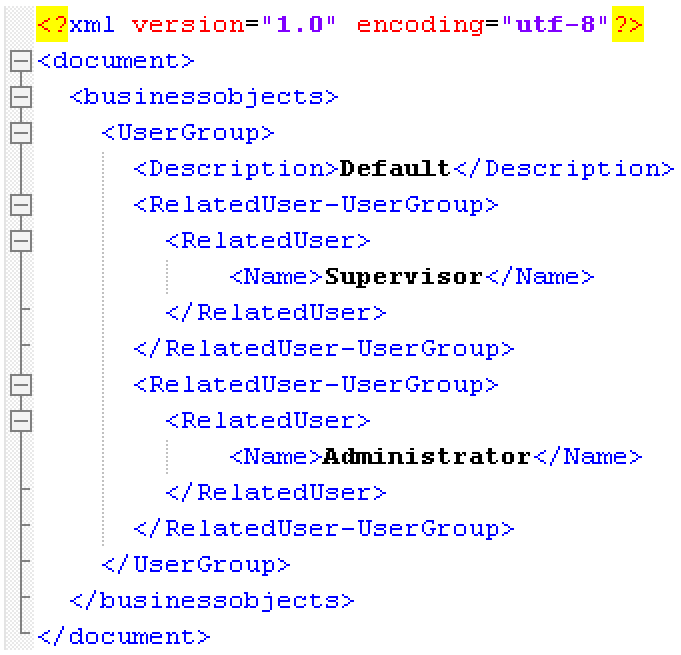
Importing/exporting linked business objects (m-to-n relation/ multiple select free field links)

If you want to import linked business objects, you can only do so if the business object link is also specified in the XML.

The following case outlines the import of User Groups and their respective users. This involves three business objects:

- User Groups
- Users
- Business object link: User groups > Related users

Sample XML



```
<?xml version="1.0" encoding="utf-8"?>
<document>
  <businessobjects>
    <UserGroup>
      <Description>Default</Description>
      <RelatedUser-UserGroup>
        <RelatedUser>
            <Name>Supervisor</Name>
        </RelatedUser>
      </RelatedUser-UserGroup>
      <RelatedUser-UserGroup>
        <RelatedUser>
            <Name>Administrator</Name>
        </RelatedUser>
      </RelatedUser-UserGroup>
    </UserGroup>
  </businessobjects>
</document>
```

The linked business object (RelatedUser-UserGroup) has two reference fields: user group and related users. In the XML sample, user group is configured as super and related user is configured as sub. This list provides an overview of all user groups and their related users.

For an example of importing linked business objects, see Importing business objects with m-to-n relation. For an example of exporting multiselect free field links, see Exporting MSFF/M-to-N links.

To avoid errors while importing, the **Part of Search key?** field must be set to **Yes** on at least one field in M-to-N business objects.

Importing/exporting maintenance activity definition schedules

To export maintenance activity definition schedules, you must select the right scheduled business objects and configure them all to retrieve all various schedule types:

- Annual schedule
- Condition-based schedule
- Condition-based schedules – library
- Daily schedules
- Deactivated frequency in schedules
- Fixed completion
- Hourly schedules
- Monthly schedules with fixed week
- Monthly schedules with fixed week & day
- Predictive-based schedules
- Single date schedules
- Weekly schedules

When updating a schedule for a maintenance activity definition, the existing schedule is deleted and will be replaced by the new schedule.

Mark imported fields

In Planon Enterprise Talk, you can mark records imported previously so that you can easily know which records were not imported this time.

Importing on a reference date

The start date specified in the XML is overruled by the date specified on the Enterprise Talk document. This has implications for two specific cases for Enterprise Talk:

- Business objects containing life cycles

  If your import file contains multiple life cycles, only a single life cycle (the last one in the XML) will be imported with the date as specified on the Enterprise Talk document.

- Business objects containing reference date aware data

  If your import file contains multiple occurrences of a reference data aware business object, only one occurrence (the last one in the XML) will be imported with the date as specified on the Enterprise Talk document.

- The start date of a contract can only be adjusted when the contract is not active. If you cannot change a value in the UI, you cannot change it by using Enterprise Talk.

- If in a space, values must be updated via the **Space usage** BO and the start date of a space usage cannot be adjusted in the UI, it can neither be done by using Enterprise Talk.

See also: Importing on a reference date.

Importing a business object containing life cycles

If you want to import a business object, for example a contract, including all the historic changes made to this contract in the past, you can import it including its life cycles.

If you want to import contract life cycles, the reference date on the **Documents** level should be empty.

If you import a contract with life cycles, the **Planon calculation start date** (PlanonBeginDate - the date from which the contract is managed in Planon) is set to the value specified in the import file, if specified. If, for a new contract, this field is still empty (also no default is configured) it is set to the start date of the contract (value of the **Start date** (BeginDate) field). The **Date effective** (ActualBeginDate) will get the value of the **Start date** (BeginDate) if it is not set in the import file.

Import file: Namespace

When the import file contains the Enterprise Talk life cycle namespace, the life cycles with their life cycle dates as specified in the Enterprise Talk namespace are imported. Including the namespace to the **<businessobjects>** tag indicates for Planon that the import file contains life cycles:

```
<businessobjects xmlns:talk='http://www.planonsoftware.com/V1/EnterpriseTalk'>
```

Import file: life cycle tags

Contract fields whose changes are saved over time are called life cycle aware fields. In the import file, these fields should be put in special tags within the tag that encloses the main business object. For example:

```
<Contract>
    <Code>1</Code>
    <Name>Contract1</Name>
    <talk:lifecycle>
      <BeginDate>2015-09-01</BeginDate>
      <OfferingContractPartyRef>0012</OfferingContractPartyRef>
    </talk:lifecycle>
```

```
<Contract>
```

This indicates that from September 1st 2015 the offering contract party is 0012.

Note that if:

- life cycle aware fields are used outside the lifecycle tags, an error is given.
- life cycle *unaware* fields (such as **Code**) are used within the life cycle tags, a warning will be logged but they will be imported.

- You must specify a BeginDate for each life cycle in the import file and also map it to the BeginDate field in the field mapping in the Enterprise Talk **Definition details**.
- When life cycle aware BOs are imported and no reference date set in the Talk document, the BeginDate should be available in all life cycle aware BOs (in a separate life cycle tag), even when the BO is only needed for reference and there is only one life cycle.
- It is not to be possible to set a life cycle aware field to part of search key.

In addition, the Suppress life cycles during in preparation status business object setting in Field definer is applied when importing contracts with Enterprise Talk. Consequently, if you want to import a new contract with history, make sure this setting is set to **No**.

The import example displays an import file containing life cycles in a contract.

Importing different UDBO types in a single run

It is possible to add multiple types of user defined business objects (UDBOs) in a single run. Since the import process needs to know the type of UDBO for doing an import, this information needs to be included in the XML (if necessary after a transformation) and in the import definition.

Example

BO **Person** has two user defined business objects:

- ‘UsrEmployee’
- ‘UsrContactpersoonExtern’

In the XML (if necessary after a transformation) the tag of the element needs to be different to identify the person type:

```
<?xml version='1.0' encoding='UTF-8'?>

<document>

  <businessobjects>

    <UsrEmployee>

 <Code>000101</Code>

 <LastName>Albert</LastName>
```

```
<Initials>A.A.</Initials>

<FacilityNetUsername>AAAlbert</FacilityNetUsername>

   </UsrEmployee>

   <UsrContactpersoonExtern>

<Code>000102</Code>

<LastName>Brown</LastName>

<Initials>B.B.</Initials>

<FacilityNetUsername>BBBrown</FacilityNetUsername>

<AddressRef>001</AddressRef>

   </UsrContactpersoonExtern>

 </businessobjects>

</document>
```

The Enterprise Talk definition needs to have two business object definitions:

1. Create an Enterprise Talk definition.
2. Add an import document.
3. Add two business object definitions:

a. One for UsrEmployee (XML node UsrEmployee)
b. One for UsrContactpersoonExtern (XML node UsrContactpersoonExtern)

4. For both UDBO definitions, add the fields as listed in the sample XML above (assumption: an address with code 001 exists in Planon).
5. For both UDBO definitions, make the Code field 'Part of the search key'.

This is only an example, this can be any other (combination of) field(s).

6. Run the Enterprise Talk definition.

   Two persons, with different UDBO type, will be added to Planon.

It is possible to update (not insert) persons with different user-defined business object (UDBO) types in a single run using only one business object definition.

Since the persons are already in Planon, the XML and the Enterprise Talk definition do not need the UDBO type information.

Updating different UDBO types in a single run

It is possible to update (not insert) persons with different user defined business object (UDBO) types in a single run using only one business object definition.

Since the persons are already in Planon, the XML and Enterprise Talk definition do not need the UDBO type information.

After first adding the persons in the import example, you can now update the person's first names by using the following XML:

```
<?xml version='1.0' encoding='UTF-8'?>

<document>

  <businessobjects>

 <Person>

    <Code>000101</Code>

    <FirstName>Alex</FirstName>

 </Person>

 <Person>

    <Code>000102</Code>

    <FirstName>Bert</FirstName>

 </Person>

  </businessobjects>

</document>
```

Complete the following steps to update the existing data:

1. Create an Enterprise Talk definition.
2. Add an import document.
3. Add a business object definition based on BO Person with XML node Person.
4. Add the Code and FirstName field to the business object definition.
5. Make the Code field ‘Part of the search key’.

This can be any other (combination of) field(s).

6. Run the definition.

   The two existing persons (based on different UDBO types) will be updated.

Adding BIM import data

If you want to add **BIM field mappings** (BIMFieldMappings) and/or **Linked business objects** (BIMLinkedBO):

- **BIM - linked properties** (BIMLinkedProperty),
- **BIM - linked floors** (BIMLinkedFloor),
- **BIM - linked spaces** (BIMLinkedSpace),
- **BIM - linked assets** (BIMLinkedAsset),
- **BIM - linked building elements** (BIMLinkedBuildingElement),

the **BIM source definition** (BIMSourceDefinition) should be added in a separate XML element.

The **BIM XML node** field (BIMXMLNodeRef) in **Linked business objects** (BIMLinkedBO) and **BIM field mappings** (BIMFieldMappings) can only be added when the **BIM source definition** (BIMSourceDefinitionRef) is added as a **Part or ID** field.

The **BIM XML node** (BIMXMLNodeRef) is often not unique, therefore a subelement needs to be created in order to uniquely identify it. In Planon ProCenter , the field/pop-up takes the **BIM source definition** (BIMSourceDefinitionRef) into account automatically.

XML sample

```
<BIMSourceDefinition>

 <BIMSourceFile>BIM/BIM_Templates/BIMCobie_1P2FL3SP9ASSET_16.xml</
BIMSourceFile>

 <Code>XYZCobieLT</Code>

 <Name>Cobie Lite for XYZ</Name>

</BIMSourceDefinition>

<BIMLinkedProperty>

 <BIMSourceDefinitionRef>XYZCobieLT</BIMSourceDefinitionRef>

 <Name>Properties</Name>

 <BIMXMLELEMENT>

  <BIMSourceDefinitionRef>XYZCobieLT</BIMSourceDefinitionRef>

  <CompositeCode>Facility</CompositeCode>

 </BIMXMLELEMENT>

</BIMLinkedProperty>
```

Archiving/Dearchiving a business object

It is possible to archive/dearchive a business object using the **Archive** field.
Enterprise Talk does not dearchive subbusiness objects of a hierarchical business object.

In Planon ProCenter , when a business object with subbusiness objects is dearchived, the subbusiness objects cannot be saved if any field is set to mandatory in the meanwhile. This is because the subbusiness object no longer complies to the business logic. If this happens, a pop-up is displayed to correct the data.

Exporting

The following sections describe the export features of Enterprise Talk:

- Exporting with or without a reference date
- Exporting system pick lists
- Exporting MSFF/M-to-N links

Exporting with or without a reference date

It is possible to create an export taking into account the reference date. You will only export business objects that exist on the reference date or take a snapshot of the data of a life cycle aware business object on that date.

If you do not specify a reference date for export, all life cycle data of a business object is exported. When exporting on a reference date, everything that is active on the reference date and later, for example space usages, will be exported.

Exporting a business object with all life cycles to CSV/Excel is not supported. Please always provide a reference date when exporting contract data to CSV/Excel.

The export example displays an output containing life cycles in a contract.

Exporting system pick lists

System pick list can only be exported. The display value of the system pick list is dependent on the language of the user conducting the export. A user in Germany will get to see German names, and a user in France will see French names.

Exporting MSFF/M-to-N links

Exporting a multiple select free field links is similar to exporting M-to-N links. In your field configuration you must include the link to the picklist whose values you want to export. For an export sample, refer to Exporting an MSFF (M-to-N) link.

Exporting Spaces and Moves Scenario Planning

In **Space and Move Scenario Planning** TSI, the **Floor** field in the **Scenario Drawing Mapping** business object must be exported as a separate business object because the floor lookup is unique only per property. The following example displayed a sample XML output.

Example:

```
<SpaceAndMoveScenarioPlanningProject>
  <Code>ScenarioTestProject</Code>
  <ProposedMoveDate>2017-02-08</ProposedMoveDate>
  <SpaceMoveScenario>
    <CADImportDefinitionRef>01</CADImportDefinitionRef>
    <Code>Scenario_1</Code>
    <ScenarioDrawingInBox>http://localhost:8070/webdav/SMSP/In</ScenarioDrawingInBox>

    <ScenarioDrawingFloorMapping>
      <ScenarioDrawing>Scenario_1.dwg</ScenarioDrawing>
      <PropertyRef>14</PropertyRef>
        <Floor>
          <Code>03</Code>
          <PropertyRef>14</PropertyRef>
        </Floor>
      </ScenarioDrawingFloorMapping>
    </SpaceMoveScenario>
</SpaceAndMoveScenarioPlanningProject>
```

Using AWS S3 Buckets

Enterprise Talk can be configured for importing data to/exporting data from AWS S3 buckets. In order to use this feature, some configuration is required.

Take the following steps to enable Enterprise Talk to work with AWS S3 buckets.

1. Create an AWS account.
2. Generate an S3 bucket policy for the bucket used for Enterprise Talk.
3. Enable a lifecycle rule for the bucket to clean up multiload requests.
4. Configure your external data storage

After completing these configuration steps, you can use S3 buckets with Enterprise Talk.

Create an AWS account

1. Go to Amazon Web Services Sign In and click Create a new AWS account.
2. Complete the procedure to create your account.

Generate the S3 bucket policy

By specifying a bucket policy, you grant permissions for the bucket and the objects in it.

To use Enterprise Talk with S3, the Planon user needs to have sufficient permissions to read from and write objects to the bucket. Below is a sample policy.

1. Fill out the correct values (in the following example these are represented by the values in between angle brackets).

```
{
  "Version": "2012-10-17",
  "Id": "<policy_id>",
  "Statement": [
    {
      "Sid": "Allow access to Planon API user",
      "Effect": "Allow",

      "Principal": {

          "AWS": "arn:aws:iam::<account_id>:user/<user_name>"

      },

      "Action": "s3:*",

      "Resource": [

          "arn:aws:s3:::<bucket_name>/*",

          "arn:aws:s3:::<bucket_name>"

      ]

    }

  ]

}
```

For more information on creating an S3 bucket policy, see

Enable a lifecycle rule

S3 offers the possibility to manage your objects' lifecycle by defining lifecycle rules. This enables you to customize the retention policy of objects in your bucket and to optimize your storage costs.

Enterprise Talk uploads objects to S3 as part of a so-called multipart upload request. To ensure that no incomplete parts are left behind, we recommended that you define a lifecycle rule on the S3 bucket as described below.

1. In your AWS Management Console, go to Management > Lifecycle and click Add lifecycle rule.

   The Lifecycle rule wizard appears.

2. Enter a name and scope and click Next.
3. In the Expiration step, select Clean up incomplete multipart uploads and specify when this should happen (number of days after starting the upload). Click Next to continue.
4. Review your configuration and click Save.

You have enabled a lifecycle rule to clean up your bucket.

For more information about configuring lifecycle policies, see http://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-lifecycle.html

Configuring your external data storage

It is possible to exchange data with an external data store by using Enterprise Talk. Currently, we only support amazon S3 buckets; the configuration is done in Planon's **External data storage** TSI. Here, you specify the connection between Planon and Amazon Web Service (AWS), and the buckets that you want to use.

1. On the Data storage credentials level, click Add AWS credentials.
2. Fill out the credentials for connecting with AWS.

For Key and Secret, enter the access key and secret of your AWS account.

3. Specify an Expiration date. You can schedule a task to notify someone when this date is approaching.
4. Go to the Data storage location level and add an S3 location.

For more information on the fields on this level, see Data storage location fields.

5. Click Save.

You have completed configuring your AWS connection, and you are now ready to use it.

Using S3 buckets

This section describes how to configure Enterprise Talk for using S3 buckets. Enterprise Talk can use S3 buckets for both import and export.

This section only describes the procedure and fields that are specific to AWS S3. Using AWS S3 buckets affects the location of the files, the Enterprise Talk functionality remains the same.

1. Create an Import-export definition.
2. Fill out a Code and Name. In the Data storage location field, select the external data storage definition that you created.
3. Go to Documents level and enter your data source. The data source path is relative to the data storage location linked to the import-export definition. That is, the S3 bucket and (if specified) the root prefix.

If you select to move the file after completing the process, specify the Post-process file location (relative to the external data storage location).

Note that:

- Transformation files should be placed in the bucket; all subsequent references are relative to the bucket.
- If a location does not exist in the bucket, it will simply be created.

- Enterprise Talk does not validate file paths, this is left to the user's responsibility.
- The configuration of the **External data storage** can be exported using **Configuration Transfer**.

Examples - import

Importing data into a single business object

To provide an example of importing data from an XML file into a single business object.

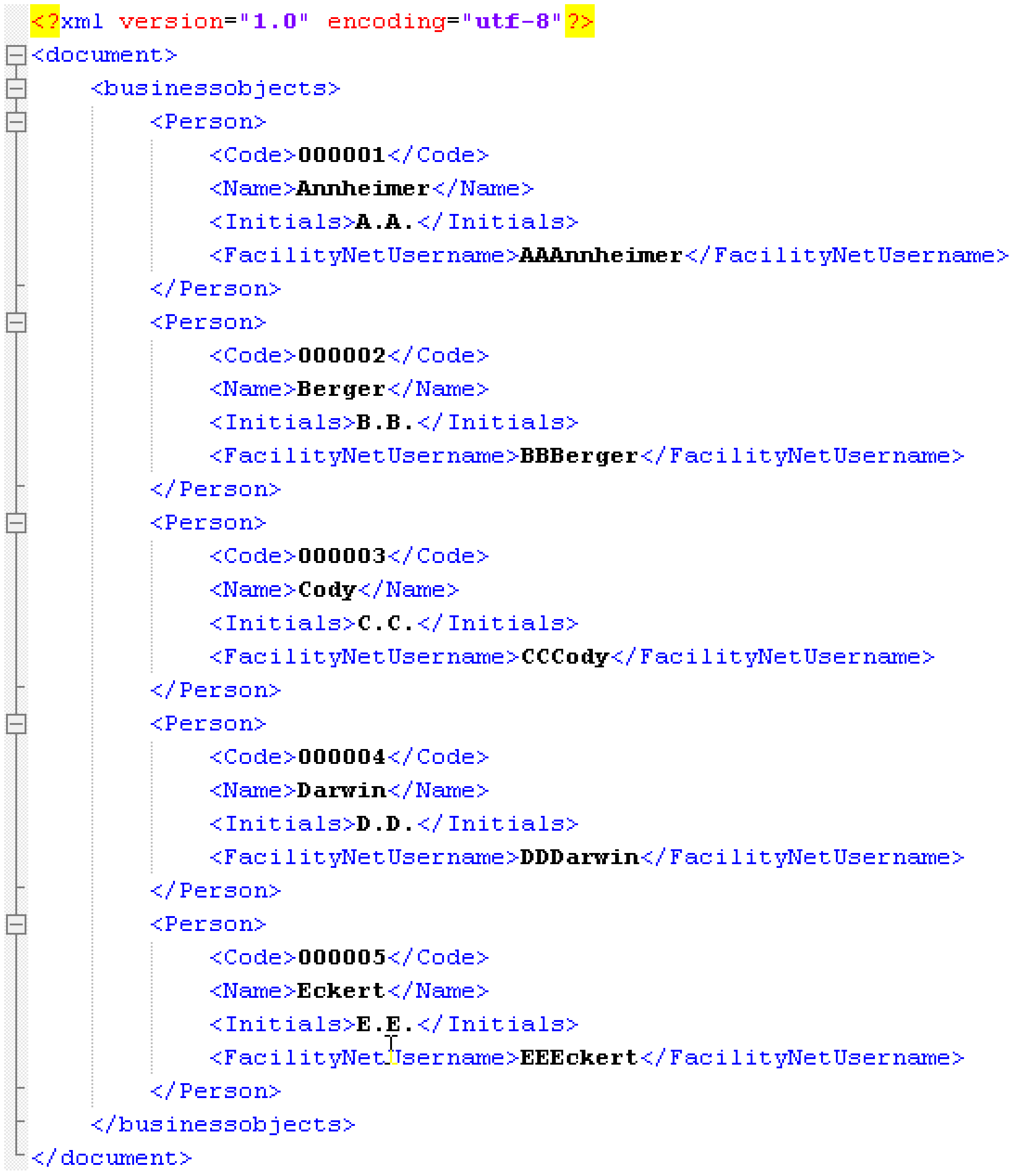
This example assumes an import definition is available and is configured correctly.

Let’s assume you want to import a list of staff members from an XML file into Planon ProCenter . On the **Definition details** level, your import definition must list the fields whose data you want to import.
You want to import the following fields:

Node name in XML	Field name in Planon ProCenter
Code	Code
Name	Surname
FaclityNetUserName	FacilityNet user name
Initials	Initials

By selecting the field name and providing the corresponding XML node name, you create a mapping so that Planon ProCenter can import the value in the correct field.

Import file


```
<?xml version="1.0" encoding="utf-8"?>
<document>
    <businessobjects>
        <Person>
            <Code>000001</Code>
            <Name>Annheimer</Name>
            <Initials>A.A.</Initials>
            <FacilityNetUsername>AAAnnheimer</FacilityNetUsername>
        </Person>
        <Person>
            <Code>000002</Code>
            <Name>Berger</Name>
            <Initials>B.B.</Initials>
            <FacilityNetUsername>BBBerger</FacilityNetUsername>
        </Person>
        <Person>
            <Code>000003</Code>
            <Name>Cody</Name>
            <Initials>C.C.</Initials>
            <FacilityNetUsername>CCCody</FacilityNetUsername>
        </Person>
        <Person>
            <Code>000004</Code>
            <Name>Darwin</Name>
            <Initials>D.D.</Initials>
            <FacilityNetUsername>DDDarwin</FacilityNetUsername>
        </Person>
        <Person>
            <Code>000005</Code>
            <Name>Eckert</Name>
            <Initials>E.E.</Initials>
            <FacilityNetUsername>EEEckert</FacilityNetUsername>
        </Person>
    </businessobjects>
</document>
```

After import, the list in Personnel TSI is amended accordingly.

Adding field as super

Importing into multiple business objects – Field as sub

To provide an example of importing nested data from an XML file into multiple business objects.

This example assumes that an import definition is available and is configured correctly.

Enterprise Talk > **Definitions** > select the definition and, on the action menu, click **Run**

Let’s assume you want to import a list of staff members and the property to which they are linked from an XML file into Planon ProCenter . On **Definition details** level, your import definition must list the fields for which you want to import data.

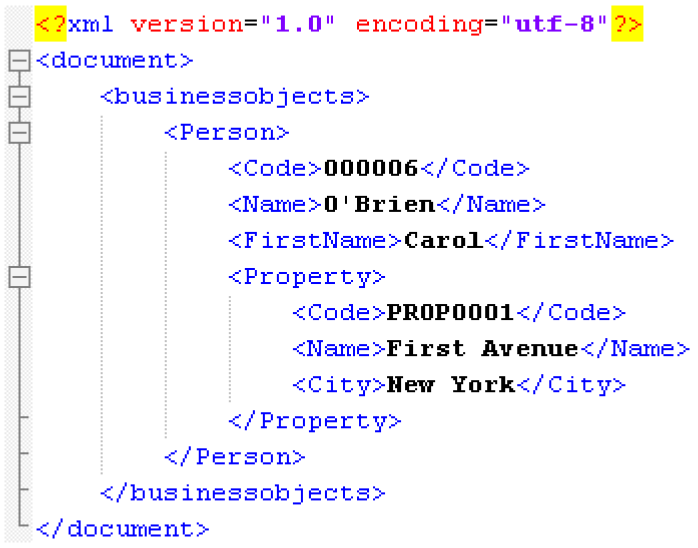
The XML file contains data fields from two business objects:

- Properties
- Personnel

Therefore, on the **Business object definitions** selection level, both business objects must be defined.
You want to import the following fields:

	Node name in XML	Field name in Planon ProCenter
Property		
	Code	Code
	Name	Name
	City	City
Person		
	Code	Code
	Name	Surname
	FirstName	First name

Import file



```
<?xml version="1.0" encoding="utf-8"?>
<document>
    <businessobjects>
        <Person>
            <Code>000006</Code>
            <Name>O'Brien</Name>
            <FirstName>Carol</FirstName>
            <Property>
                <Code>PROP0001</Code>
                <Name>First Avenue</Name>
                <City>New York</City>
            </Property>
        </Person>
    </businessobjects>
</document>
```

Since your main business object is Person, in the XML, the Property data is included as a lower level to the Person data. Consequently, on the **Definition details** level, the Property reference must be added as sub to the Person business object.

Before import

In Personnel TSI, the property First Avenue is not in the list of properties. The count of properties is 61.

After import

The property, First Avenue is added to the list of properties. The count of properties is incremented by one and is now: 62. In the data section, the **Property code**, **Name** and **City** fields are filled out.

If you descend to the **Personnel** selection level, the staff member is listed with **Code**, **Surname**, and **Property** filled out.

Composite business object

Nested data

Adding field as sub

Importing into multiple business objects – Field as super

To provide an example of importing nested data from an XML file into a multiple business objects

This example assumes an import definition is available and configured correctly.

Enterprise Talk > **Definitions** > select the definition and, on the action menu, click **Run.**

Let's assume you want to import a list of staff members and their related properties from an XML file into Planon ProCenter . On the **Definition details** level, your import definition must list the fields whose data you want to import.

The XML file contains data fields from two business objects:

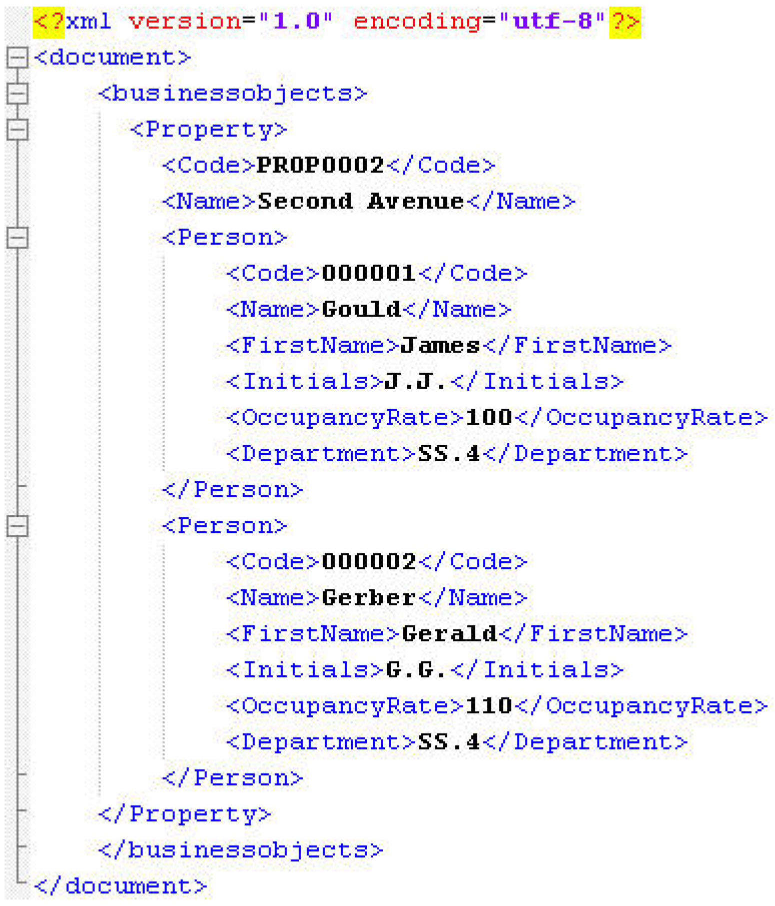
- Personnel
- Properties

Therefore, on **Business object definitions** selection level, both business objects must be defined.

You want to import the following fields:

	Node name in XML	Field name in Planon ProCenter
Person		
	Code	Code
	Name	Surname
	OccupancyRate	Occupancy rate
	Department	Department
	FirstName	First name
	Initials	Initials
Property		
	Code	Code
	Name	Name

Import file



```xml
<?xml version="1.0" encoding="utf-8"?>
<document>
    <businessobjects>
      <Property>
        <Code>PROP0002</Code>
        <Name>Second Avenue</Name>
        <Person>
            <Code>000001</Code>
            <Name>Gould</Name>
            <FirstName>James</FirstName>
            <Initials>J.J.</Initials>
            <OccupancyRate>100</OccupancyRate>
            <Department>SS.4</Department>
        </Person>
        <Person>
            <Code>000002</Code>
            <Name>Gerber</Name>
            <FirstName>Gerald</FirstName>
            <Initials>G.G.</Initials>
            <OccupancyRate>110</OccupancyRate>
            <Department>SS.4</Department>
        </Person>
    </Property>
    </businessobjects>
</document>
```

Since your main business object is Property, in the XML, the Property data is included at a higher level to the Person data. Consequently, on the **Definition details** level, the Property reference must be added as super to the Person business object.

As visible in above import file reference fields, in this case the **Department** field, can also be populated directly by using the lookup value. In general the code of a referenced BO can be used. For department the lookup value is the codegroup field since the business object is hierarchical and the code not unique.

Result

Before import, Gerald G.G. Gerber and James J.J. Gould were not in the list of personnel. After import, both persons are added to the list of personnel and the count is updated.

In addition, in Personnel TSI > Properties, the property Second Avenue is now added to the list.

Composite business object

Nested data

Adding field as sub

Adding field as super

Importing business objects with hierarchical levels

It is possible to add business objects that have hierarchical relations to each other. Departments have a hierarchical structure, below an example on how to import them.

XML:

```
<?xml version='1.0' encoding='utf-8'?>

       <Department>

        <Code>T01</Code>

        <Description>Test 01</Description>

        <ParentCode></ParentCode>

       </Department>

       <Department>

        <Code>T02</Code>

        <Description>Test 02</Description>

        <ParentCode>T01</ParentCode>

       </Department>

       <Department>

        <Code>T03</Code>

        <Description>Test 03</Description>

        <ParentCode>T01</ParentCode>

       </Department>
```

You can add a level in the xml by using a XSLT transformation:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

    xmlns:xs="http://www.w3.org/2001/XMLSchema" version="2.0">
```

```
                 <xsl:template match="/">
                     <document>
                     <businessobjects>
               <xsl:for-each select="document/businessobjects/Department">
                <Department>
                 <Code><xsl:value-of select="Code" /></Code>
                 <Description><xsl:value-of select="Description" /></Description>
                 <Parent>
                  <Code><xsl:value-of select="ParentCode" /></Code>
                 </Parent>
                </Department>
               </xsl:for-each>
              </businessobjects>
              </document>
              </xsl:template>
             </xsl:stylesheet>
```

Within a Talk import document two business objects definitions need to be created both on BO **Departments**. One for the parent department and one for the sub department. This is because the departments have two levels (in this example).

Configuration

Two department BOs need to be added in the definition with a different XML node name, one for the parent department and one for the child. The parent Department BO only requires the **Code** field. The child Department BO requires a **Code**, a **Description** and a reference to the **Parent**.

Importing business objects with m-to-n relation

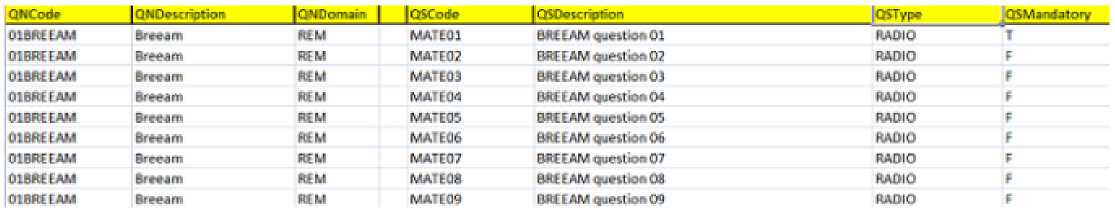
To avoid errors while importing, the **Part of Search key?** field must be set to **Yes** on at least one field in M-to-N business objects.

This example describes an import of an excel file with questionnaire and it’s (single select) questions.

Questions can be linked to multiple questionaires so there is an m-to-n link table (QuestionnaireQuestion) between the questionaire and the question.

This QuestionnaireQuestion m-to-n table has, next to links to questionaire and question also a field for the Sequence number.

The excel file is flat and looks like this:



QNCode	QNDescription	QNDomain	QSCode	QSDescription	QSType	QSMandatory
01BREEAM	Breeam	REM	MATE01	BREEAM question 01	RADIO	T
01BREEAM	Breeam	REM	MATE02	BREEAM question 02	RADIO	F
01BREEAM	Breeam	REM	MATE03	BREEAM question 03	RADIO	F
01BREEAM	Breeam	REM	MATE04	BREEAM question 04	RADIO	F
01BREEAM	Breeam	REM	MATE05	BREEAM question 05	RADIO	F
01BREEAM	Breeam	REM	MATE06	BREEAM question 06	RADIO	F
01BREEAM	Breeam	REM	MATE07	BREEAM question 07	RADIO	F
01BREEAM	Breeam	REM	MATE08	BREEAM question 08	RADIO	F
01BREEAM	Breeam	REM	MATE09	BREEAM question 09	RADIO	F

The excel should be transformed to create a three level xml file. This is because three Business objects needs to be populated.

```
<?xml  version="1.0" encoding="utf-8"?>
  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/
              1999/XSL/Transform">
     <xsl:output method="xml" />
     <xsl:template match="Questionnaire">
        <Questionnaire>
       <QNCode><xsl:value-of select="QNCode" /></QNCode>
       <QNDescription><xsl:value-of select="QNDescription" />
           </QNDescription>
       <QNDomain><xsl:value-of select="QNDomain" /></QNDomain>
              <QN_QS_LINK>
           <Question>
          <QSCode><xsl:value-of select="QSCode" /></QSCode>
      <QSDescription><xsl:value-of select="QSDescription" />
        </QSDescription>
      <QSMandatory><xsl:value-of select="QSMandatory" />
           </QSMandatory>
      <QSType><xsl:value-of select="QSType" /></QSType>
       </Question>
       <SequenceNumber><xsl:value-of select="SequenceNumber" />
           </SequenceNumber>
```

```
            </QN_QS_LINK>

        </Questionnaire>

      </xsl:template>

    </xsl:stylesheet>
```

The document wil have 4 workers:

- ExcelReaderWorker
- XSLWorker
- XMLToPOJOConvertor
- PlanonWriterWorker

And three business object definitions:

- Questionnaire
- Question
- QN_QS_LINK

QN_QS_LINK refers to the QuestionnaireQuestion m-to-n link table. The questions are (in this example) all added as singleSelectQuestions.

Next to adding fields to the Questionnaire and Question, fields should be added to the QuestionnaireQuestion business object. To this BO two reference fields should be added: one to the Question and one to the Questionnaire business objects and the SequenceNumber field.

Importing user-defined links

It is possible to import user-defined links or multiple select fields using an Enterprise Talk definition.

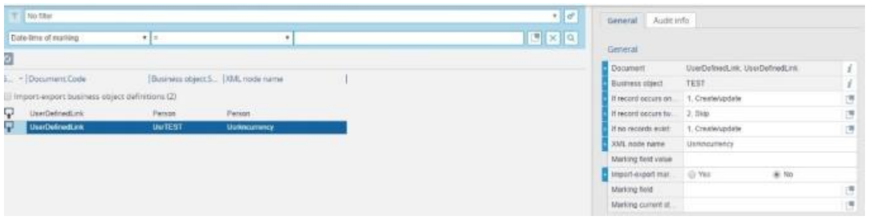
Sample XML:

```
<businessobjects>

    <Usrkncurrency> (this is the name of the link created in field definer with “links”)

     <UserCodeCodeRef>AFN</UserCodeCodeRef> (the content of the picklist you want
 to add as multiple select)

       <Person>

         <Code>P000000472</Code>

       </Person>
    </Usrkncurrency>

  </businessobjects>
```

The following procedure, with reference to the above example, shows how to link a user-defined pick list item to an existing person:

Procedure

1. Create a pick list with at least one item to call on first. In the example, the pick list Usrkncurrency is created with one item AFN to call on.
2. Create a definition. Use the above sample XML and make the required changes.
3. Create two Import-export business object definitions. In the example, one is created for the pick list Usrkncurrency and the other for the Person business object.
4. The Usrkncurrency pick list business object must be set to Create/ Update in all fields as shown in the following screenshot:



No filter

Date-time of marking =

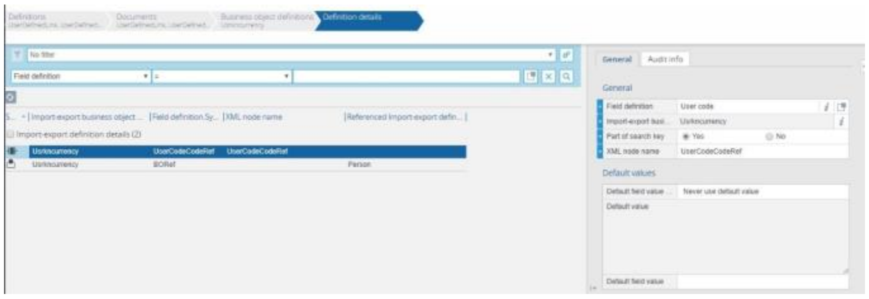
Document.Code	Business object S...	XML node name
Import-export business object definitions (2)		
UserDefinedLink	Person	Person
UserDefinedLink	UsrTEST	Usrkncurrency

General | Audit info

General

Document	UserDefinedLink, UserDefinedLink
Business object	TEST
If record occurs on...	1, Create/update
If record occurs fo...	2, Skip
If no records exist	1, Create/update
XML node name	Usrkncurrency
Marking field value	
Import-export mal...	Yes [✓ No]
Marking field	
Marking current st...	

5. The Person business object must be set to Skip in all fields as shown in the following screenshot:



Definitions UserDefinedLink, UserDefined... > Documents UserDefinedLink, UserDefined... > Business object definitions Usrkncurrency > Definition details

No filter

Field definition =

Import-export business object...	Field definition Sy...	XML node name	Referenced Import-export defin...
Import-export definition details (2)			
Usrkncurrency	UserCodeCodeRef	UserCodeCodeRef	
Usrkncurrency	BORef		Person

General | Audit info

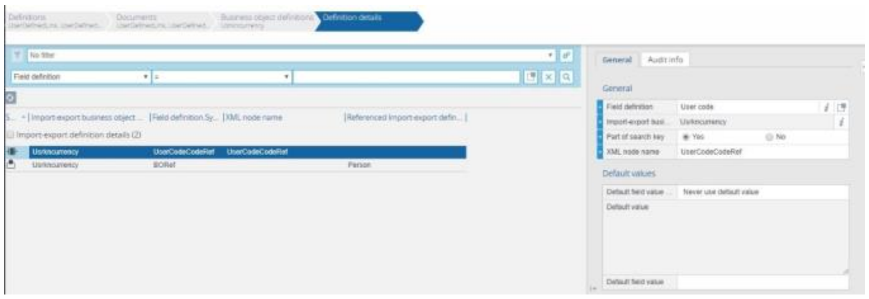
General

Field definition	User code
Import-export busi...	Usrkncurrency
Part of search key	[✓ Yes] No
XML node name	UserCodeCodeRef

Default values

Default field value ...	Never use default value
Default value	
Default field value	

6. On Usrkncurrency, create a sub field definition that refers to the **Person** business object using the BORef field and set the field Part of search key to Yes.
7. On the Usrkncurrency pick list, create a standard Import-export field definition that refers to theUserCodeCodeRef, and set the field Part of search key to Yes as shown in the following screenshot:



S...	Import-export business object...	Field definition Sy...	XML node name	Referenced import-export defin...
Import-export definition details (2)				
	Usrkncurrency	UserCodeCodeRef	UserCodeCodeRef	
	Usrkncurrency	BORef		Person

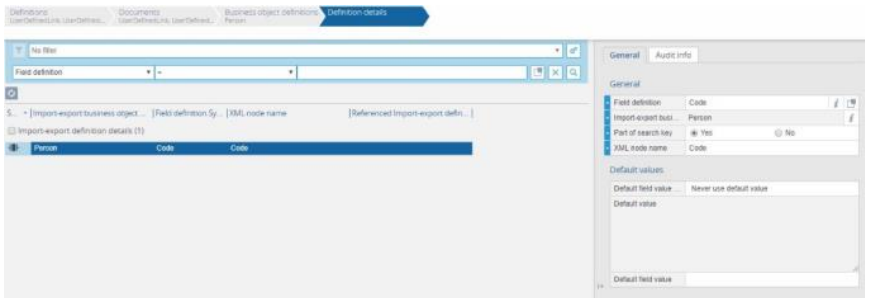
General

Field definition	User code
Import-export busi...	Usrkncurrency
Part of search key	(●) Yes ( ) No
XML node name	UserCodeCodeRef

Default values

Default field value ...	Never use default value
Default value	
Default field value	

8. For the Person business object, link the business object definition Code and set the field Part of search key to Yes as shown in the following screenshot:



S...	Import-export business object...	Field definition Sy...	XML node name	Referenced import-export defin...
Import-export definition details (1)				
	Person	Code	Code	

General

Field definition	Code
Import-export busi...	Person
Part of search key	(●) Yes ( ) No
XML node name	Code

Default values

Default field value ...	Never use default value
Default value	
Default field value	

9. Add the pick list link on the Action pane > Links of the Layout of the Person to whom you want to link to. For example, Employee > Layout > usrkncurrency.
10. Run the Enterprise Talk defintion.

After running the definition, the referenced pick list item is added to the person.

You can also import UserMtoNAssociation (user-defined links) via Enterprise Talk. In the Planon application, you can import both multiple select picklists and standard Planon links via Enterprise Talk.

Importing assets and asset locations

Assets can be linked to one or multiple locations. If the asset is of type 'simple' and you are only importing properties and spaces and no other fields from location assignment, you can define the space and property on the asset itself and do not need to specify location assignments.

For multiple assets, or if you want to use other fields from the location assignment, proceed as follows:

1. Create two business object definitions: one on BO **Assets** and one on BO **Asset locations**.

2. Select the **Asset locations** business object definition and go to **Definition details** level. Link the asset reference field (as super) to the Asset BO and add the **Property** field.
3. Return to the **Business object definitions** level and select the **Assets** business object definition. Make sure this definition contains the asset **Code**, **Name** and **Item Group** fields.
4. Import the XML.

Import XML sample

```
<businessobjects>

      <Assets>

       <Name>Pool car - Citroen</Name>

       <Code>CAR006</Code>

       <ItemGroupRef>07.01</ItemGroupRef>

       <InventoryLocationAssignment>

        <PropertyRef>14</PropertyRef>

       </InventoryLocationAssignment>

      </Assets>

     </businessobjects>
```

The asset and asset location will be updated.

Importing communication logs

Communication log data is stored in two business objects and, in addition, it is related to a specific type of business object.

It is possible to import communication logs using these two business objects and the referenced business object. Therefore, in the import document, three business object definitions need to be created:

- UsrWerkorderAlgemeen
- CommunicationLog
- ComLogBaseOrder

For the **UsrWerkorderAlgemeen** business object, add the **Description** and **OrderNumber** fields.

For the **CommunicationLog** business object, add the **Code**, **BeginDate** and **Comment** fields.

For the **ComLogBaseOrder** business object, add a field as super referencing the **UsrWerkorderAlgemeen**, and add a field as sub referencing the **CommunicationLog** business object.

For communication logs, uploading secure documents is not yet supported.

Import XML samples

Order linked to communication log

An order with number 140.00 should exist in Planon.

```
<?xml version='1.0' encoding='utf-8'?>

<document>

  <businessobjects>

     <UsrWerkorderAlgemeen>

        <OrderNumber>140.00</OrderNumber>

        <Description>standard workorder 2</Description>

        <ComLogBaseOrder>

           <CommunicationLog>

                <CommentString>Communication log imported</CommentString>

                <Code>1</Code>

                <BeginDate>2015-04-14T13:03:00</BeginDate>

           </CommunicationLog>

        </ComLogBaseOrder>

  </UsrWerkorderAlgemeen>

</businessobjects>

</document>
```

Communication log linked to order

An order with number 629.00 should exist in Planon.

```
<?xml version='1.0' encoding='utf-8'?>

<document>
```

```
    <businessobjects>
      <UsrCommunicationLog>
        <Comment>Some text to import</Comment>
        <BeginDate>2015-04-14T12:16:00</BeginDate>
        <ComLogBaseOrder>
          <UsrOrder>
            <OrderNumber>629.00</OrderNumber>
          </UsrOrder>
        </ComLogBaseOrder>
     </UsrCommunicationLog>
    </businessobjects>
  </document>
```

Importing workspaces

This topic describes how to import workspaces by using Enterprise Talk.

When adding or updating a workspace, you also need to include the workspace detail. If the workspace detail is not included, an error will be shown.

Typically, the space code itself is not unique, which is why space, floor and property are also added. In this case, the space, floor and property already exist so they are only used as search fields and the BOs themselves are skipped. The import document should therefore have five business elements that are interlinked.

Field configuration

Business Object level	Import policy	Field (system name)	Field as …	Part of search key?
Property	Skip			
		Code		Yes
Floor	Skip			
		Code		Yes
		Property	Sub	Yes

Business Object level	Import policy	Field (system name)	Field as …	Part of search key?
Space	Skip			
		Code		Yes
		Name		
		Floor	Sub	Yes
WorkSpace-Details	Create/ update			
		BeginDate		No
		Space	Sub	Yes
		WorkSpace	Super	Yes
WorkSpace	Create/ update			
		Name		No
		Code		Yes
		BeginDate		Yes
		AvailableArea		No

Import XML sample

```
<?xml version='1.0' encoding='utf-8'?>

<document>

 <businessobjects>

  <WorkSpace>

   <Code>100</Code>

   <Name>Workspace</Name>

   <BeginDate>2023-4-15</BeginDate>

   <AvailableArea>1</AvailableArea>

   <WorkSpaceDetails>

    <Space>
```

```
        <Code>1.42</Code>

        <Floor>

         <Code>01</Code>

         <Property>

          <Code>14</Code>

         </Property>

        </Floor>

        <Name>WS1.42.a</Name>

       </Space>

       <BeginDate>2023-4-15</BeginDate>

      </WorkSpaceDetails>

     </WorkSpace>

    </businessobjects>

   </document>
```

After import: a fixed workspace starting on 15-4-2023 is added to Planon for property 14, floor 1, space 1.42.

Importing occupancies

This topic describes how to import occupancies by using Enterprise Talk.

When adding an occupancy it is not sufficient to add a reference to the workspace since the workspace is reference-date-aware and unique with its workspace details. As also described in Importing workspaces, workspace details should always be added to the workspace to uniquely identify it.

When importing an occupancy for a workspace that has multiple time-aware workspace details (records created by ending/resuming a workspace), you need to set the reference date on the Enterprise talk document to import an occupancy. By doing so, it will be possible to import a single time-aware workspace on the given reference date.

The following example displays a sample XML for adding an occupancy (so including the workspace and workspace detail). Make sure to correctly set all **Part of ID** fields in the Enterprise Talk definition.

Import XML sample

```
<?xml version='1.0' encoding='utf-8'?>
```

```
<document>
 <businessobjects>
  <Occupancy>
   <CostCentreRef>030000</CostCentreRef>
   <DepartmentRef>03.01</DepartmentRef>
   <EndDate></EndDate>
   <Rate>100.00</Rate>
   <PersonPositionRef>56</PersonPositionRef>
   <RequiredArea>16.00000000000000</RequiredArea>
   <BeginDate>2016-01-01</BeginDate>
   <WorksOnFriday>True</WorksOnFriday>
   <WorksOnMonday>True</WorksOnMonday>
   <WorksOnSaturday>False</WorksOnSaturday>
   <WorksOnSunday>False</WorksOnSunday>
   <WorksOnThursday>True</WorksOnThursday>
   <WorksOnTuesday>True</WorksOnTuesday>
   <WorksOnWednesday>True</WorksOnWednesday>
   <PersonRef>032</PersonRef>
   <WorkSpace>
    <Code>WS_2.23_19</Code>
    <BeginDate>2004-04-01</BeginDate>
    <AvailableArea>1</AvailableArea>
    <WorkSpaceDetails>
     <BeginDate>2004-04-01</BeginDate>
     <Space>
      <Code>2.23</Code>
      <Floor>
       <Code>02</Code>
       <Property>
        <Code>14</Code>
       </Property>
      </Floor>
     </Space>
    </WorkSpaceDetails>
   </WorkSpace>
  </Occupancy>
 </businessobjects>
</document>
```

After import: an occupancy starting on 01-01-2016 is added to Planon for workspace WS_2.23_19, space 2.23, floor 02, property 14.

Importing on a reference date

If the reference date is set on **Documents** level, the date specified there overrules the date in the XML.

Import document: reference date is set to 07-09-2015.

Input XML file:

```
<businessobjects>
    <UsrLeaseContract>
      <Code>AA</Code>
      <Name>AA</Name>
      <IsPaymentContract>True</IsPaymentContract>
      <ContractCategoryRef>01</ContractCategoryRef>
      <BeginDate>2015-09-01</BeginDate>
      <OfferingContractPartyRef>0012</OfferingContractPartyRef>
        <ContractIdCounterpart/>
    </UsrLeaseContract>
</businessobjects>
```

Because the start date specified in the XML is overruled by the date on the Enterprise Talk document, the result is a contract with a start date of 07-09-2015.

Importing a contract with life cycles

The following example describes what happens when you import the following import file containing life cycles. The reference date on **Documents** level is not specified.

Import file

```
<?xml version='1.0' encoding='UTF-8'?>

<document>

  <businessobjects xmlns:talk='http://www.planonsoftware.com/V1/EnterpriseTalk'>

    <UsrLeaseContract>

      <Code>A</Code>

      <Name>AA</Name>

      <IsPaymentContract>True</IsPaymentContract>

      <ContractCategoryRef>01</ContractCategoryRef>

      <ActualBeginDate>2015-08-09</ActualBeginDate>

      <talk:lifecycle>

        <BeginDate>2020-09-01</BeginDate>

        <OfferingContractPartyRef>0012</OfferingContractPartyRef>

      </talk:lifecycle>

      <talk:lifecycle>

        <BeginDate>2020-09-02</BeginDate>

        <OfferingContractPartyRef>0009</OfferingContractPartyRef>

      </talk:lifecycle>

      <LeaseContractLine>

        <Code>AAL </Code>

        <AmountUnitOfTimeRef>3</AmountUnitOfTimeRef>

        <PropertyRef>14</PropertyRef>

        <Name>AALines</Name>

        <DaysPaymentBeforeClosing>2</DaysPaymentBeforeClosing>

        <IsPaymentContractLine>True</IsPaymentContractLine>
```

```
          <PaymentFrequencyDate>2015-08-09</PaymentFrequencyDate>

          <ContractLinePeriod>Unknown</ContractLinePeriod>

          <talk:lifecycle>

            <PaymentFrequency>1 Months</PaymentFrequency>

            <Amount>10.00</Amount>

            <BeginDate>2015-08-09</BeginDate>

          </talk:lifecycle>

          <talk:lifecycle>

            <PaymentFrequency>1 Months</PaymentFrequency>

            <Amount>11.00</Amount>

            <BeginDate>2020-09-02</BeginDate>

          </talk:lifecycle>

        </LeaseContractLine>

      </UsrLeaseContract>

    </businessobjects>

  </document>
```

Field configuration

Business Object level	Import policy	Field	Field as …	Part of search key?
usrLeaseCon-tract	Create/ update			
		Code		Yes
		Name		No
		BeginDate		No
		IsPaymentContract		No
		ContractCategoryRef		No
		OfferingContractPartyRef		No

Business Object level	Import policy	Field	Field as …	Part of search key?
LeaseContrac-tLine	Create/ update			
		Code		Yes
		ContractRef	Super	Yes
		Amount		No
		IsPaymentContractLine		No
		PaymentFrequency		No
		PropertyRef		No
		AmountUnitOfTimeRef		No
		DaysPaymentBefore-Closing		No
		BeginDate		No
		ContractLinePeriod		No
		PaymentFrequencyDate		No
		Name		No

The result will be a lease contract with two life cycles:

Code/ description	BeginDate	EndDate	OfferingContract-PartyRef
A-AA	2020-09-01	2020-09-01	0012
	2020-09-02	<empty>	0009

Including a lease contract line with two life cycles:

Code/ description	BeginDate	EndDate	Amount
AA-AALines	2015-08-09	2020-09-01	10
	2020-09-02	<empty>	11

Importing a business object without a unique key field (Budget)

In order to import the budget, the budget category needs to be identified by Code and Financial year, because the same code can exist over many financial years.

Field configuration

Business Object level	Import policy	Field	Field as …	Part of search key?
Budget	Create/ update			
		Code		No
		BudgetSize		No
		Description		No
		BudgetCategory	Sub	No
BudgetCateg-ory	Skip			
		Code		Yes
		FinancialYear-Ref		Yes

Input file

```
<?xml version="1.0" encoding="UTF-8"?>

<document>

   <businessobjects>

    <Budget>

      <BudgetSize>10000.00</BudgetSize>

      <Code>15</Code>

      <Description>Maintenance budget</Description>

 <BudgetCategory>

          <Code>0002</Code>
    <FinancialYearRef>2011</FinancialYearRef>

      </BudgetCategory>

   </Budget>

  </businessobjects>

</document>
```

The result will be a budget “15-Maintenance budget” linked to the budget category 0002 of financial year 2011.

Importing business objects - applying inheritance

When importing nested business objects, the business object hierarchy needs to defined in the import XML.

When you import a sub-order of an existing order without specifying the business object relationship, the import will fail. Consider the following example:

```
<UsrInternalWorkOrder>

 <ParentOrderRef>2743.00</ParentOrderRef>

</UsrInternalWorkOrder>
```

In this example, because the business object hierarchy is not specified, the inheritance business logic is not applied and the import will fail because mandatory fields are not inherited from the existing order.

By specifying the business object relationship in your XML import definition, you ensure that the inheritance business logic is applied:

```
<UsrInternalWorkOrder>

  <OrderNumber>2743.00</OrderNumber>

  <UsrInternalWorkOrder>

    <Description>Sub for 2743.00</Description>

  </UsrInternalWorkOrder>

</UsrInternalWorkOrder>
```

In this example, the business object relationship is specified by nesting the XML. Consequently, the business logic is applied and the import will work.

Importing the SlaMtoNContractLine reference field (SLA contract line link)

In the ServiceLevelAgreementScope (**SLA scope**) and SLAContractLineCosts (**SLA contract line costs**) business objects, Enterprise Talk can be used to populate the combination of the contract line and the linked SLA in the SlaMtoNContractLine (**SLA contract line link**) field. This field is a reference to the ServiceLevelAgreementMtoNContractLine M-to-N BO. The SLA field and the contract line fields are read-only, they cannot be populated.

For identification you may be required to set the Enterprise Talk reference date.

The SlaMtoNContractLine field can be populated in Enterprise Talk by using the following XML structure (example with SLA scope):

```
<ServiceLevelAgreementScope>
 <Code>SC1</Code>
 <Name>Scope1</Name>
 <PropertyRef>1</PropertyRef>
 <ServiceLevelAgreementMtoNContractLine>
  <SLAContractLine>
   <Code>CL1</Code>
   <ContractRef>Contract1</ContractRef>
  </SLAContractLine>
  <ServiceLevelAgreement>
   <Code>CL1</Code>
   <Name>ContractLine1</Name>
  </ServiceLevelAgreement>
 </ServiceLevelAgreementMtoNContractLine>
</ServiceLevelAgreementScope>
```

Examples - export

Exporting data from multiple business objects

You are able to export data of multiple business objects into an xml file. For instance you would want to export a property with all persons located in this property.

Procedure

1. Create an export definition.
2. Create an export document for the export definition. In the Data source box, refer to the XML file to create. (PlanonReaderWorker and XMLWriter workers are added)
3. Create a business object definition for the Properties business object, make it the main business object and add a filter to retrieve the required property
4. Add the required fields of property to the definition details
5. Create a business object definition for the Person business object
6. Add the required fields of person (LastName and FirstName) to the definition details. Include the Property field as a field as super and make the reference to the property definition
7. Run the definition

```
<?xml version='1.0' encoding='utf-8'?>
<document>
 <header>
  <creation-date>2014-03-20T13:59:45</creation-date>
  <user>ADMINISTRATORUK</user>
  <EE-version state="Production">201311.0.4.0P</EE-version>
  <metadataversion>B_FOEE_2013110000_FB11280</metadataversion>
  <database>MSSQL</database>
  <measurementunit>METER</measurementunit>
  <ExportDefinitionCode>EXP_PROP_PERS</ExportDefinitionCode>
  <ExportDocumentCode>P_P</ExportDocumentCode>
  <SequenceOfDocument>1</SequenceOfDocument>
 </header>
 <businessobjects>
  <Property>
   <Name>Columbus Campus</Name>
   <Address>Columbus Lane</Address>
   <City>London</City>
   <PostalCode>N6 5TR</PostalCode>
   <UsrEmployee>
    <FirstName>Ben</FirstName>
    <LastName>Humphrey</LastName>
   </UsrEmployee>
   <UsrEmployee>
    <FirstName>Peter</FirstName>
    <LastName>Pool</LastName>
   </UsrEmployee>
  </Property>
 </businessobjects>
</document>
```

Exporting communication logs

Communication log data is stored in two business objects and, in addition, it is related to a specific type of business object.

It is possible to export communication logs using these two business objects and the referenced business object. In the export document three business object definitions need to be created:

- UsrCommunicationLog
- UsrOrder
- ComLogBaseOrder

For the **UsrCommunicationLog** add the fields you want to export. For example the **Comment** and **BeginDate** field.

For the **UsrOrder** also add the fields you want to export. For example the **OrderNumber** field.

For the **ComLogBaseOrder** business object, two reference fields should be added:

- BORef (field as sub) to BO UserOrder
- CommunicationLogRef (field as super) to BO UsrCommunicationLog

The following XML will be created:

```
<?xml version='1.0' encoding='utf-8'?>
      <document>
      <header>...</header>
      <businessobjects>
       <UsrCommunicationLog>
        <Comment></Comment>
        <BeginDate>2011-07-08T11:29:00</BeginDate>
       </UsrCommunicationLog>
       <UsrCommunicationLog>
        <Comment></Comment>
        <BeginDate>2011-07-08T11:29:00</BeginDate>
       </UsrCommunicationLog>
       <UsrCommunicationLog>
        <Comment>test comlog</Comment>
        <BeginDate>2014-04-28T13:40:00</BeginDate>
        <ComLogBaseOrder>
         <UsrOrder>
          <OrderNumber>629.00</OrderNumber>
         </UsrOrder>
        </ComLogBaseOrder>
       </UsrCommunicationLog>
      </businessobjects>
```

```
</document>
```

- Next to communication logs of orders, also other communication logs are listed.
- By using a transformation you can remove the communication logs of other business objects.
- When com log data of other business objects are needed instead of the **Order** business object, these business objects should be added to the definition.

Export to CSV

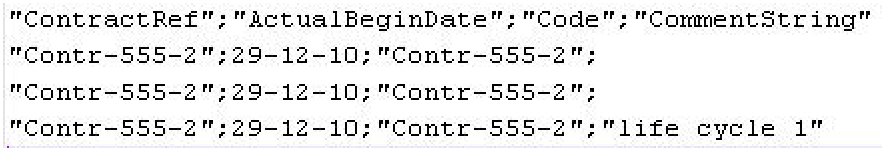
XML is the default format for export. To be able to export Planon ProCenter data to CSV, data transformation is required.

Planon ProCenter features a default CSVWriterWorker that can read and transform XML into CSV.

The export process comes with a default FileXMLWriterWorker that transforms a POJO to XML. However, to be able to further transform the XML to CSV, you must replace the default FileXMLWriterWorker with the POJOToXMLStringWorker.

Typically, the order in the XML is used for displaying the fields. If you want a specific order, in the **Additional parameters** box, specify the order by using the “columns=” parameter followed by the field names.

When running this export definition, the Planon ProCenter data is written to a CSV output file:



```
"ContractRef";"ActualBeginDate";"Code";"CommentString"
"Contr-555-2";29-12-10;"Contr-555-2";
"Contr-555-2";29-12-10;"Contr-555-2";
"Contr-555-2";29-12-10;"Contr-555-2";"life cycle 1"
```

Exporting floors

Floors are uniquely identified by a combination of property and floor. In order to export floors, create two business object definitions:

- Space
- Floor

Make **Floor** field as sub of **Space**.

Output XML sample

```
<businessobjects>
```

```
    <Space>
      <Name>Installation Space</Name>
      <PropertyRef>41008</PropertyRef>
      <Floor>
        <Code>0</Code>
        <PropertyRef>41008</PropertyRef>
      </Floor>
    </Space>
  </businessobjects>
```

Exporting the default answer of a question

Exporting a default answer of a question (Questionnaires TSI) is possible despite that the **Default answer** field is not a database field.

To carry out these steps, you require:

- A configured questionnaire in Questionnaires TSI.
- This questionnaire should have a question configured for and linked to it.
- The **Default answer** field should contain a value.

For the purpose of this example, the **Code** field of the linked question is ***101***, and the **Default value** field contains the value: ***This is the default value of a question***.

- In **Enterprise Talk**, you have already configured a definition and an export document.

Including a question's default answer in your export.

Procedure

1. Go to the Business object definitions level and add an export business object definition.
2. Make the following settings:

a. Select the Base question business object.
b. Select Yes for the Main business object field
c. Select your export document.
d. In the Filter field, open the filter and specify Code = 101 (to uniquely identify your question).
e. On the action menu, click Business object definition fields and put the Default value XML field In use.

f. Save your changes.

3. Go to the Definitions level and click Run on the action menu.

   Your export will be created in the location that you specified. Note that the DefaultValueXML tag contains the exported default answer.

```
<?xml version='1.0' encoding='utf-8'?>
<document>
 <header>
  <creation-date>2015-04-09T16:24:46</creation-date>
  <user>Planon</user>
  <EE-version state="Production">201503.0.0.0</EE-version>
  <metadataversion>FB10001</metadataversion>
  <database>Oracle</database>
  <measurementunit>METER</measurementunit>
  <ExportDefinitionCode>101</ExportDefinitionCode>
  <ExportDocumentCode>101</ExportDocumentCode>
  <SequenceOfDocument>4</SequenceOfDocument>
 </header>
 <businessobjects>
  <TextQuestion>
   <DefaultValueXml>This is the default value of a question</DefaultValueXml>
   <Code>101</Code>
  </TextQuestion>
 </businessobjects>
</document>
```

Exporting on a reference date

This example explains how to export all data of a contract, a life cycle aware business object. In the example, the accepting party of a contract has a changed. The BO fields selected for export are the following:

- Code
- Name
- Start date contract (BeginDate)
- Accepting party

The fields **Code** and **Name** are not life cycle aware; if these fields change value, they change for the entire contract because their changes are not tracked over time. The **Accepting party** on the other hand is a life cycle field. When you do not specify a reference date, all its values over time are exported. If you want to know when a change was applied, you need to include the field **Begindate**.

Each life cycle is captured in its own tag and the resulting XML is as follows:

```
<businessobjects xmlns:talk="http://www.planonsoftware.com/V1/EnterpriseTalk">
    <UsrLeaseContract>
      <Code>0001</Code>
      <Name>Airport Boulevard - Carbon Smart (out)</Name>
      <talk:lifecycle>
        <BeginDate>2008-01-01</BeginDate>
        <AcceptingContractPartyRef>0002</AcceptingContractPartyRef>
      </talk:lifecycle>
      <talk:lifecycle>
        <BeginDate>2015-09-04</BeginDate>
        <AcceptingContractPartyRef>0005</AcceptingContractPartyRef>
      </talk:lifecycle>
    </UsrLeaseContract>
</businessobjects>
```

Exporting a business object with all life cycles to CSV/Excel is not supported. Please always provide a reference date when exporting contract data to CSV/Excel.

Exporting an MSFF (M-to-N) link

Exporting an multiple select link is similar to exporting M-to-N links.

Example

Person Mease has drivers licences A and B through link UsrLicenseCodes to picklist UsrCode_LICENSE (LICENSE).

1. In your Enterprise Talk definition, add three business objects:
   - Person
   - UsrLicenseCodes
   - UsrCode_LICENSE
2. Add the fields to these business objects as shown in the following table:

Business Object	Field	Field as...	Part of search key?
Person	LastName		Yes
UsrLicenseCodes	Business object (BORef) to BO Person (link)	Super	Yes
	User Code (UserCodeCodeRef) to BO UsrCode_LICENSE (link)	Sub	Yes
UsrCode_LICENSE	Code		Yes
	Name		No

3. Once you have finished configuring, run your definition.

   The XML result will look as follows:

```
<Person>

 <LastName>Mease</LastName>

 <UsrLicenseCodes>

  <UsrCode_LICENSE>

   <Code>A</Code>

   <Name>Motor cycle license</Name>

  </UsrCode_LICENSE>

 </UsrLicenseCodes>
 <UsrLicenseCodes>
  <UsrCode_LICENSE>
   <Code>B</Code>
   <Name>Car license</Name>
  </UsrCode_LICENSE>
 </UsrLicenseCodes>
</Person
```

Examples - data transformation

The following topics provide examples of data transformation

XSLT

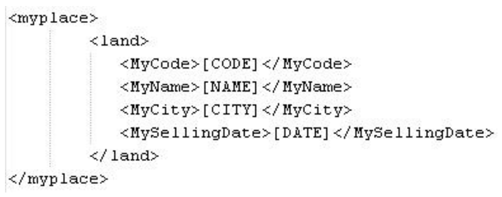
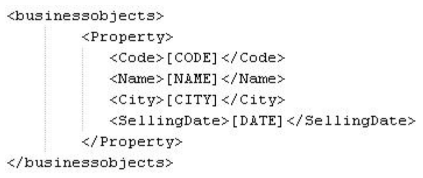
Data transformation example (import)

If you want to import data from any system into Planon ProCenter , data transformation may be required.

If, for example, you want to import an XML data file containing property data into Planon ProCenter , you can use an XSLT file to convert the file data into a format that is understood by Planon ProCenter .

Example

The following table lists the source format and the format as expected by Planon ProCenter .

Source XML	Planon ProCenter **XML**
	

```
<myplace>
        <land>
           <MyCode>[CODE]</MyCode>
           <MyName>[NAME]</MyName>
           <MyCity>[CITY]</MyCity>
           <MySellingDate>[DATE]</MySellingDate>
        </land>
</myplace>
```

```
<businessobjects>
        <Property>
           <Code>[CODE]</Code>
           <Name>[NAME]</Name>
           <City>[CITY]</City>
           <SellingDate>[DATE]</SellingDate>
        </Property>
</businessobjects>
```

In order to import the source XML file, the data in the file needs to be converted to make it compatible with Planon ProCenter .

Procedure

1. Create an import definition.
2. Create a document for the import definition. In the Data source box, refer to the source XML file.
3. Create a business object definition for the Properties business object.
4. On the Import-export worker step, on the action menu, click Add to add the FileXSLTTransformation worker (for example). Adjust the sequence so the worker will start in the correct order.

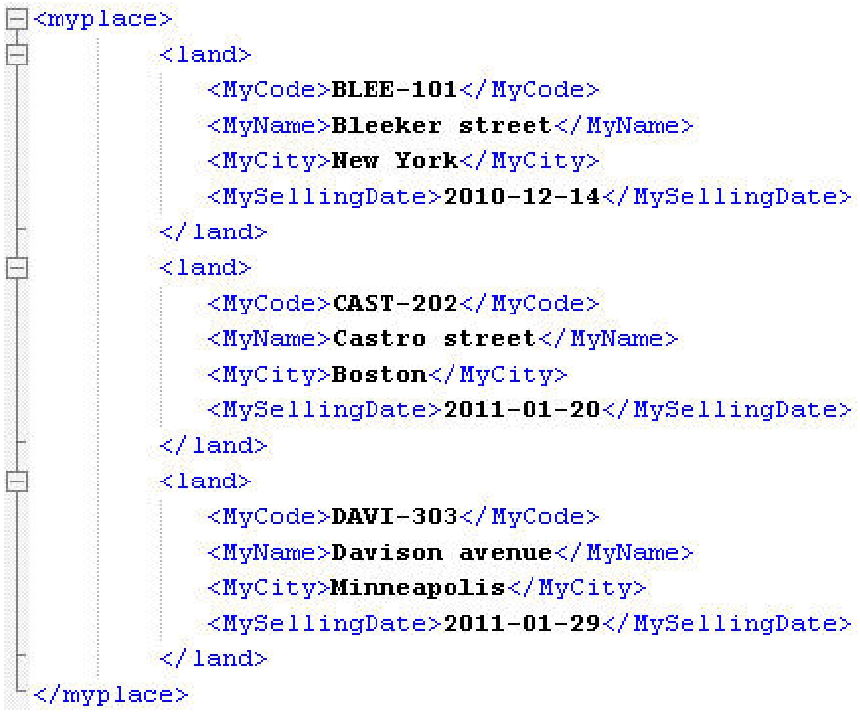
For Planon ProCenter to read the data in the source XML file, the data transformation must first be processed. Consequently, ensure the sequence of processing this XSLT transformation is before reading the file.

5. For the FileXSLTTransformation worker, in the Additional parameters box, provide the file specifics:

```
in=C:\path\to\source_xml_file.xml,

out=C:\path\to\source_converted.xml,

xslfile=C:\path\to\XSLT_transformation_file.xslt
```

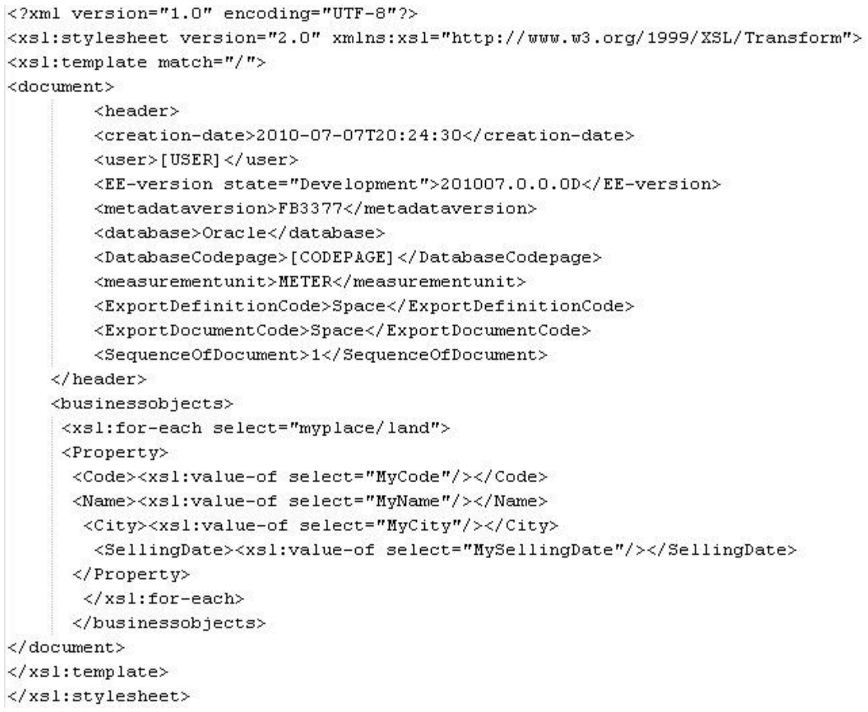
6. Save and run the definition.

Source XML file



```
<myplace>
        <land>
            <MyCode>BLEE-101</MyCode>
            <MyName>Bleeker street</MyName>
            <MyCity>New York</MyCity>
            <MySellingDate>2010-12-14</MySellingDate>
        </land>
        <land>
            <MyCode>CAST-202</MyCode>
            <MyName>Castro street</MyName>
            <MyCity>Boston</MyCity>
            <MySellingDate>2011-01-20</MySellingDate>
        </land>
        <land>
            <MyCode>DAVI-303</MyCode>
            <MyName>Davison avenue</MyName>
            <MyCity>Minneapolis</MyCity>
            <MySellingDate>2011-01-29</MySellingDate>
        </land>
</myplace>
```

XSLT file



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<document>
        <header>
        <creation-date>2010-07-07T20:24:30</creation-date>
        <user>[USER]</user>
        <EE-version state="Development">201007.0.0.0D</EE-version>
        <metadataversion>FB3377</metadataversion>
        <database>Oracle</database>
        <DatabaseCodepage>[CODEPAGE]</DatabaseCodepage>
        <measurementunit>METER</measurementunit>
        <ExportDefinitionCode>Space</ExportDefinitionCode>
        <ExportDocumentCode>Space</ExportDocumentCode>
        <SequenceOfDocument>1</SequenceOfDocument>
    </header>
    <businessobjects>
     <xsl:for-each select="myplace/land">
     <Property>
      <Code><xsl:value-of select="MyCode"/></Code>
      <Name><xsl:value-of select="MyName"/></Name>
       <City><xsl:value-of select="MyCity"/></City>
        <SellingDate><xsl:value-of select="MySellingDate"/></SellingDate>
      </Property>
       </xsl:for-each>
      </businessobjects>
</document>
</xsl:template>
</xsl:stylesheet>
```

Data transformation example (export)

If, for instance, you want to export properties and the exported xml is not formatted as desired you can transform the xml to another with a different layout.

Example standard export Talk of properties:

```
<?xml version='1.0' encoding='utf-8'?>

    <document>

     <header>

      <creation-date>2014-03-20T14:37:33</creation-date>

      <user>ADMINISTRATORUK</user>

      <EE-version state="Production">201311.0.4.0P

            </EE-version>

      <metadataversion>B_FOEE_2013110000_FB11280

            </metadataversion>

      <database>MSSQL</database>

      <measurementunit>METER</measurementunit>
```

```
   <ExportDefinitionCode>EXP_PROP</ExportDefinitionCode>
   <ExportDocumentCode>exp-prop</ExportDocumentCode>
   <SequenceOfDocument>1</SequenceOfDocument>
  </header>
  <businessobjects>
   <Property>
    <Address>Columbus Lane</Address>
    <City>London</City>
    <Country>United Kingdom</Country>
    <Name>Columbus Campus</Name>
    <IsArchived>False</IsArchived>
   </Property>
   <Property>
    <Address>Columbus Square</Address>
    <City>London</City>
    <Country>United Kingdom</Country>
    <Name>Columbus Square</Name>
    <IsArchived>False</IsArchived>
   </Property>
   <Property>
    <Address>Columbus Lane</Address>
    <City>London</City>
    <Country>United Kingdom</Country>
    <Name>Columbus building</Name>
    <IsArchived>False</IsArchived>
   </Property>
  </businessobjects>
 </document>
```

In the final xml the header should be removed as well as the country field and the IsArchived field should contain a different code. Like this:

```
<?xml version="1.0" encoding="UTF-8"?>
         <document xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <businessobjects>
           <property>
            <Name>Columbus Campus</Name>
            <Address>Columbus Lane</Address>
            <City>London</City>
            <IsArchived>0</IsArchived>
           </property>
           <property>
            <Name>Columbus Square</Name>
            <Address>Columbus Square</Address>
            <City>London</City>
            <IsArchived>0</IsArchived>
           </property>
           <property>
            <Name>Columbus building</Name>
            <Address>Columbus Lane</Address
            ><City>London</City>
            <IsArchived>0</IsArchived>
           </property>
          </businessobjects>
         </document>
```

Following xsl file is used to do the transformation:

```
<?xml version="1.0" encoding="UTF-8"?>
      <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
          xmlns:xs="http://www.w3.org/2001/XMLSchema" version="2.0">

          <xsl:template match="/">

              <document>

                  <businessobjects>

                      <xsl:for-each select="document/

                              businessobjects/Property">

                          <property>

         <Name>

                        <xsl:value-of select="Name"/>

         </Name>

         <Address>

                         <xsl:value-of select="Address"/>

         </Address>   <City>

                         <xsl:value-of select="City"/>

         </City>

         <xsl:if test="IsArchived='False'">

             <IsArchived>0</IsArchived>

         </xsl:if>

            </property>

                      </xsl:for-each>

                  </businessobjects>

              </document>

          </xsl:template>

      </xsl:stylesheet>
```

Procedure

1. Create an export definition.
2. Create a document for the export definition. In the Data source box, refer to the XML file to export.
3. Create a business object definition for the Properties business object. (see also the example ‘exporting data from multiple business objects’)

4. Create another export document. In the Data source box, refer to the transformed XML file.
5. On the Import-export worker step, on the action menu, click Add to add an XSLT file as a new worker. Add the parameters as specified in the worker specification in section Using standard workers.
6. On the same step remove the PlanonReaderWorker and XMLWriter worker.
7. Run the definition

Data transformation example (Transform datetimes)

When Talk exports a date time field it is, depending on the type, exported in the following format:

YYYY-MM-DDTHH:mm:SS, for example 2013-11-13T14:12:10

One would like the date to be displayed as DD-MM-YYYY, for example 13-11-2013

You need an export transformation. For more information, also refer to Data transformation example (export).

The original xml file:

```
<document>
    <businessobjects>
     <order>
        <startDate>2013-11-13T14:12:10</startDate>
       </order>
     </businessobjects>
  </document>
```

XSLT (xsl file)

```
<?xml version="1.0" encoding="UTF-8"?>
  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" version="2.0">
    <xsl:template match="/">
      <document>
```

```
                <businessobjects>
                    <xsl:variable name="bos" select="document/
                                               businessobjects"/>
                    <xsl:for-each select="document/businessobjects/order">
                        <order>
                            <convertedDate>
                                <xsl:value-of select="format-dateTime
                                    (startDate,'[D]-[M]-[Y]')"/>
                            </convertedDate>
                        </order>
                    </xsl:for-each>
                </businessobjects>
            </document>
        </xsl:template>
    </xsl:stylesheet>
```

Result

```
<?xml version="1.0" encoding="UTF-8"?>
    <document xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <businessobjects>
            <order>
                <convertedDate>13-11-2013</convertedDate>
            </order>
        </businessobjects>
    </document>
```

For an import one would like to transform from for example 2013-11-13T14:12:10 to 13-11-2013

XSLT (xsl file

```
<?xml version="1.0" encoding="UTF-8"?>

  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

    xmlns:xs="http://www.w3.org/2001/XMLSchema" version="2.0">

    <xsl:template match="/">

      <document>

        <businessobjects>

          <xsl:variable name="bos" select="document

                    /businessobjects"/>

          <xsl:for-each select="document/businessobjects/order">

            <order>

              <convertedDate>

      <xsl:variable name="dt" select="startDate "/>

      <xsl:value-of select="concat(substring($dt, 9, 2),

          '-',substring($dt, 6, 2),'-',substring($dt, 1, 4),

          'T00:00:00')"/>

              </convertedDate>

            </order>

          </xsl:for-each>

        </businessobjects>

      </document>

    </xsl:template>

  </xsl:stylesheet>
```

Using standard workers

Workers may be specific to the import process, export process, or they can also be bidirectional.

Standard workers

The following sections list the workers that are standard available in Planon.

For an overview of all workers and their path, see Standard Worker Properties.

Import workers

This section lists the workers that can be applied to the import process.

FetchFilesWorker

`nl.planon.morpheus.pnworkers.server.xmlworkers.FetchFilesWorker`

This worker can read multiple import files. If required, it should be the fist worker in the list (according to the sequence).

The worker will read the files in the order sorted on the file modification date/time of the operating system. The oldest files are read first.

For more information on reading multiple files during import, refer to Reading multiple files

XMLReaderWorker

`nl.planon.morpheus.pnworkers.server.xmlworkers.XMLReaderWorker`

This worker uses a STaX parser to extract main business objects one at a time when the execute API is called by the Worker Manager. This component reads data from the XML files for importing data into Planon ProCenter .

The output of this Worker is an XML string corresponding to the main business object in the XML.

XMLToPOJOWorker

`nl.planon.morpheus.pnworkers.server.xmlworkers.XMLToPOJOWorker`

The Workers associated with the Planon Writer and Reader work with a POJO. This system worker converts an XML string to a POJO that is required by the Planon Workers.

PlanonDataWriterWorker

`nl.planon.morpheus.pnworkers.server.planonwriterworker.PlanonDataWriterWorker`

This worker interacts on the POJO corresponding to the source's main business object that is passed to it and persists this in Planon ProCenter . The writer also applies the import policy defined in the definition to load the details into Planon ProCenter .

With the **Approval** feature in Planon, you can configure that important changes must always be approved by a second person before the changes are actually implemented. See Approvals.

When importing data via **Enterprise Talk** or **SDI configuration**, it is possible to bypass the triggering of the approval process for changes that normally require approval. Via the **triggerApproval** parameter on the import worker, the triggering of approvals can be skipped for the data to be imported via this worker.

For the **PlanonDataWriterWorker**, in the **Additional parameters** box, add the following line:

```
triggerApproval = Yes/No
```

Possible settings:

- **triggerApproval** = **Yes**: to explicitly define that approvals are triggered when importing/updating data through this worker.
- **triggerApproval** = **No**: to skip triggering approvals when importing/ updating data through this worker. Even if an approval is required by the approval definition, data is simply changed and no approval is generated.
- If the **triggerApproval** parameter is not registered, approvals are triggered if necessary.

CSVReaderWorker

`nl.planon.morpheus.pnworkers.server.csvworkers.CSVReaderWorker`

If you need to import from a CSV file, the workers needed are:

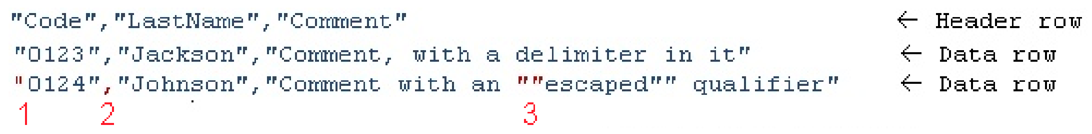
- CSVReaderWorker
- XMLToPOJOConvertor
- PlanonWriterWorker

A CSV file is converted into XML data. This component reads data from the CSV files for importing data into Planon ProCenter

The output of this Worker is an XML string corresponding to each main business object in the CSV.

An example file of CSV data is used to explain the effect of the CSVReaderWorker with the terminology used to describe its parameters.

Example CSV file



```
"Code","LastName","Comment"                                              ← Header row
"0123","Jackson","Comment, with a delimiter in it"                       ← Data row
"0124","Johnson","Comment with an ""escaped"" qualifier"                 ← Data row
 1     2                            3
```

1 = text qualifier (“)

2 = delimiter (,)

3 = an escaped text qualifier

If parameter mainElementName=Person, the resulting XML file would be:

```
  <Person>
<Code>00123</Code>
<LastName>Jackson</LastName>
<Comment>Comment, with a delimiter in it</Comment>


</Person>
<Person>
<Code>00124</Code>
<LastName>Johnson</LastName>
<Comment>Comment with an "escaped" qualifier</Comment>

</Person>
```

For a description of the parameters of the CSVReaderWorker, refer to CSVReaderWorker.

ExcelReaderWorker

`nl.planon.morpheus.pnworkers.server.excelworker.ExcelReaderWorker`

The ExcelReaderWorker reads Excel sheets (xls, xlsx) with data so as to import them into Planon ProCenter .

For a description of the parameters, refer to ExcelReaderWorker.

Example:

```
tabSheet=Persons

          mainElementName=Person

          columns=D;E;F;G;H;I;J;K;L;M;N;O;P;Q

          rowHeader=9

          rowStartData=11

          numberOfRows=50

          nrOfEmptyRowsToStop

          unitOfMeasurement=M

          columnForStopDef=D
```

The output of the ExcelReaderWorker is XML. The ExcelReaderWorker allows you to specify a cell containing the reference date using the 'referenceDateCell' parameter. There are three ways to to specify a reference date in Excel:

- Use 'referenceDateCell'

  When you specify the parameter 'referenceDateCell' and you do not specify parameter 'referenceDateElementName' a BeginDate attribute will be added containing the fixed date value from 'referenceDateCell'.

- Use 'referenceDateCell' and 'referenceDateElementName'

  In the XML a tag will be added with the name specified in the 'referenceDateElementName' the tag will be populated with the 'referenceDateCell' value. The position of this tag is by default the last exported field in the XML. If a different order is required, you can specify the order with parameter 'columns' and add the value of the 'referenceDateCell' parameter at the right location.

DBReaderWorker to read databases

```
nl.planon.morpheus.pnworkers.server.dbreaderworker.DBReaderWorker
```

This reader worker allows you to connect to a database and run SQL commands to export data and import it into Planon ProCenter, so that it becomes easy to transfer data.

- You can connect only to external databases, not make a connection to the current Planon database.
- An extra data source needs to be created for the external database. The data source should be added to the *…\Server\wildfly-*\standalone*

\configuration\standalone-full.xml file. This can be done with the Wildfly Command line interface.

For information about the Wildfy Command line interface see the Administrator's Guide.

- This extra datasource should be named <name>DS, for example Worker-DS.
- Make sure the database user can only read from the assigned database.
- In the default worker configuration, the XMLReaderWorker should be replaced by the DBReaderWorker.
- The databases supported for the DBReaderWorker are MSSQL and Oracle by default. Other databases are not supported since the corresponding drivers are not available as part of the install set.

Datasource file example

```
<datasource jndi-name="java:/jdbc/WorkerDS" pool-name="WorkerDS" enabled="true"
 spy="true" jta="false">

  <connection-url>jdbc:sqlserver://[ServerName]\SQLEXPRESS2014</connection-url>

  <driver>mssql-test</driver>

  <new-connection-sql>PLN_SET_P5_SESSION</new-connection-sql>

  <pool>

    <min-pool-size>1</min-pool-size>

    <!-- The minimum number of connections maintained in the pool. Unless <prefill> is
 true then the pool will remain empty until first use at which point the pool will be filled to the
 <minpool-size>. When the pool size drops below the <min-pool-size> due to idle timeouts the
 pool will be refilled to the <min-pool-size>. Default is 0.-->

    <max-pool-size>5</max-pool-size>

    <!-- The maximum number of connections allowed in the pool. Default is 20.-->

  </pool>

  <statement>

    <prepared-statement-cache-size>25</prepared-statement-cache-size>

    <!-- The number of prepared statements per connection to be kept open and reused in
 subsequent requests. They are stored in a LRU cache. The default is 0 (zero), meaning no
 cache.-->

  </statement>

  <timeout>

    <blocking-timeout-millis>60000</blocking-timeout-millis>
```

```
     <!-- The length of time to wait for a connection to become available when all the
 connections are checked out. Default is 30000 (30 seconds).-->

     <idle-timeout-minutes>5</idle-timeout-minutes>

     <!-- Indicates the maximum time a connection may be idle before being closed. Setting to 0
 disables it. Default is 15 minutes.-->

     <!-- Note: Incase, prefill is set to true then this will not close the idle connections-->

   </timeout>

   <validation>

     <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"/>

     <check-valid-connection-sql>select 1</check-valid-connection-sql>

     <background-validation>true</background-validation>

     <background-validation-millis>10000</background-validation-millis>

     <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter"/>

   </validation>

   <security>

     <user-name>John Doe</user-name>

     <password>Plan$QL</password>

   </security>

</datasource>

<drivers>

  <driver name="mssql-test" module="com.mssql">

    <xa-datasource-class
 transactionisolation="TRANSACTION_NONE">com.microsoft.sqlserver.jdbc.SQLServerXADataSource</
xa-datasource-class>

   </driver>

</drivers>
```

Encrypting worker parameter value

The encrypt text tool in Planon ProCenter , allows you to encrypt any text to be used as a secure value for a parameter in a custom worker in Enterprise Talk. When you enter plain text and tab out of the field, it is automatically converted into an encrypted string.

The encrypted string can be decrypted in the worker using the API from the Java docs.

BOM which can send back encrypted string by passing a regular text can be made available for all types so that custom workers which require a parameter having an excerpted string to be passed can use this feature.

To encrypt parameter value:

Procedure

1. Go to Business object definition > Import-export worker.
2. On Import-export worker menu, click Encrypt plain text. The Encrypt plain text dialog box appears.
3. Enter text in the Plain text field and click outside the field. The Encrypted text field is automatically populated with the encrypted version of the plain text.

Export workers

This section lists the workers that can be applied to the export process.

PlanonDataReaderWorker

```
nl.planon.morpheus.pnworkers.server.planonreaderworker.PlanonDataReaderWorker
```

The PlanonDataReaderWorker is used for exporting data from Planon ProCenter . It identifies the main business object and exports the required fields, associations and referenced business object details.

This worker reads a main business object as defined in the Enterprise Talk configuration, creates the POJO, and passes it back to the Worker Manager via the context.

FileXMLWriterWorker

```
nl.planon.morpheus.pnworkers.server.xmlworkers.FileXMLWriterWorker
```

A worker that writes the exported details from Planon ProCenter . It uses a STaX parser to write details into an XML file.

For a description of the parameters, refer to FileXMLWriterWorker parameters.

CSVWriterWorker

```
nl.planon.morpheus.pnworkers.server.csvworkers.CSVWriterWorker
```

If you need to export to a CSV file, the workers needed are:

- PlanonReaderWorker
- POJOToXMLStringWorker
- CSVWriterWorker

For a description of the parameters, refer to CSV WriterWorker.

Example

Parameter specification which can be used as a template.

# Main element name in the XML. If it is not specified here, it will be taken

# from the node name on the BO definition if there is only one, or it will

# default to "element".

mainElementName=

# Should the CSV have a header? yes/no, or y/n, or true/false

# Default = yes, so if you don’t specify this parameter we generate a header.

header=yes

# Names/order of the columns. Comma separated. Column names are case sensitive.

# This setting will determine the order and subset of columns that will be

# exported. If you don’t fill in this parameter, the order from the XML is used

# and all columns are exported.

columns=

# The delimiter symbol, default value is semicolon.

# Valid values are: ;/semicolon/semi and ,/comma and tab/\t and space/\s

# and any literal symbol, e.g: #

delimiter=semicolon

# Text qualifier symbol. Valid values are: '/apostrophe/apos and "/quote/quot

textQualifier=quote

# What to do if the target file already exists. Only applicable for export.

# Valid values are: cancel/stop and replace/overwrite and new/create.

# new/create will create a new file with a timestamp added to the filename.

ifFileExists=replace

# Encoding to use if it could not be detected from a Unicode Byte Order Mark.

# Valid values are names and aliases from the Character Set Registry maintained

# by IANA: http://www.iana.org/assignments/character-sets

fileEncoding=utf-8

# Should the UTF-8 Byte Order Mark be written (if fileEncoding is utf-8).

# Valid values are: yes/no, or y/n, or true/false. Only applicable for export.

writeUtf8Bom=yes

# Whether debug information should be logged for each imported/exported record.

# Valid values are: yes/no, or y/n, or true/false

debug=no

POJOToXMLStringWorker

```
nl.planon.morpheus.pnworkers.server.xmlworkers.POJOToXMLStringWorker
```

This worker converts a POJO to an XML string and is used in exporting to CSV, see CSVWriterWorker.

Bidirectional workers

This section lists the workers that can be applied to both the import as the export process.

XMLChunkXSLTTransformationWorker

`nl.planon.morpheus.pnworkers.server.xsltworkers.XMLChunkXSLTTransformationWorker`

This worker transforms an XML file using the transformation file supplied in the parameters and converts this to an XML string.

This particular worker operates on one main business object at a time. For a description of these fields, refer to XSL Transformation.

For transformation, Enterprise Talk supports XSLT 2.0 and uses Saxon transformer version Saxon-HE-9.6. This version also supports the ‘URI resolver’.

If the XSLT already provides the right input to the POJO, no further worker is required. However, in most cases, the chunk worker will require a custom Data Reader, because the default Planon Reader assumes that the <document> tag indicates the start of the document and the <businessobjects> tag indicates the main business object entries.

FileXSLTTransformationWorker

`nl.planon.morpheus.pnworkers.server.xsltworkers.FileXSLTTransformationWorker`

This worker can be used to perform an XSL transformation. This particular worker operates on an entire file.

Typically, this will be the first worker in the worker configuration. For a description of the fields that are available for this worker, refer to XSLT Transformation Worker.

Example import

```
in=\temp\import\persons.xml,
out=\temp\import\persons_trans.xml,
xslfile=\temp\import\xsl\persons.xsl
```

Example export

```
xslfile=c:\showRes\reservation.xsl,
in=c:\showRes\resRaw.xml,
out=c:\showRes\res.xml
```

Creating customized workers

You can sequentially perform various operations on the source data before the data is either imported into Planon ProCenter or exported from Planon ProCenter .

Example

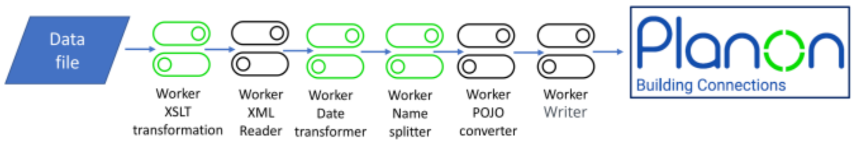
An XML file containing a set of main business objects has to be read and imported into Planon ProCenter .

- An XSLT has to be applied on the entire file to make it an XML file that the worker reading the XML file understands.
- The XML file has to be parsed and one main business object at a time needs to be extracted.
- On the XML string, the date has to be transformed to match the Planon ProCenter application.
- The name has to be split up into first name, last name, etc.
- The string has to be converted to a POJO that is understood by Planon ProCenter .

  These steps represent individual operations for which workers have to be created. Workers have to be given a sequence so that they can be executed in that order.

  In this case, the following workers can be defined:

1. XSLT Transformation worker
2. XML Reader Worker
3. Date Transformer Worker
4. Name Splitter Worker
5. POJO Convertor Worker
6. Planon Writer Worker



(A custom worker is green).

Enterprise Talk supports XSLT 2.0 and uses Saxon transformer version Saxon-HE-9.6. This version also supports the ‘URI resolver’.

The chain of workers can change based on the requirement. The workers will be determined by the nature of the source of data and its format:

All workers are Java components that take in source data, process it and return an output. The output from one worker will be the input for the next worker in the chain. A set of workers that is created for a specific purpose is called a worker bundle.

Given the appropriate Java knowledge, customers can create their own worker bundles.

For technical information about creating workers, see the Java doc. In the Java doc, all instances of *Data Import Export Manager* should be read as *Enterprise Talk*. The Java doc is available from the installer. To extract it, run the installer and select Related components > Manual installation resources > Planon Software Development Kit. After extracting the resources, you will find the Java doc in:*<your folder>/related_components/manual_installation_resources/sdk/ImportExportWorker/index-all.html*

POJO

Worker

Worker bundle

Workers chain

Creating a customized worker

- Creating customized workers requires specific Java knowledge.
- Use an Eclipse environment to build a worker bundle.

Create a new OSGI Plug-in project from the API JAR.

Planon ProCenter provides key interfaces that enable users to develop an import/export worker bundle. By using the interfaces provided by Planon ProCenter , the user is able to bring about the desired functionality. The interfaces for developing such workers are shipped as JAR archives to the end user who wants to develop the import/export worker.

The customized workers are plugged into Planon ProCenter as a worker bundle. This allows the user to develop and upgrade workers on the fly and deploy them on the application without causing any disturbance.

The following steps provide an overview of how to develop an import/export worker in Eclipse.

To compile your client extension, the worker APIs should be a bundle in your Eclipse workspace before creating an import/ export worker bundle. For this, import the PnImportExportWorker.jar file as a plug-in in the workspace first. This file is located in the ..\Distribution\ImportExportWorker folder.

Procedure

1. In Eclipse, click File > New Project. The New Project dialog box appears.
2. Expand Plug-in Development folder and select Plug-in from existing jar archives.
3. Click Next and, in the next dialog box, click Add and point to the PnImportExportWorker.jar file.
4. Click Next and provide a project name.
5. Click Finish to complete the setup. Your workspace is now ready to develop import/export workers. Now, you are ready to create a new Worker OSGI Plug-in.
6. In Eclipse, in the new workspace that is created, create a new Plug-in project. In the New Plug-in Project dialog box, enter a name and select (OSGI framework) Equinox as the target platform.

For the purpose of this procedure, the name ExampleWorker is used.

7. Click Next and then click Finish. Now, the skeleton bundle is created with the activator for the bundle in place. Next, modify the manifest.mf file to refer to the required packages.
8. Open the manifest.mf file. This file is in the same location as is your project.

   In the import package, include the following package.

   nl.planon.hades.dataimportexport.worker.interfaces

   Your manifest.mf should resemble the following sample:

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: ExampleWorker Plug-in
Bundle-SymbolicName: ExampleWorker
Bundle-Version: 1.0.0
Import-Package:
 org.osgi.framework,
 nl.planon.hades.dataimportexport.worker.interfaces
```

The examples used here are simplified by excluding the Import-Package versions. It is recommended to version the imports with the compatible version range of the import. This will help in detecting compatibility of the workers, when an upgrade has been performed.

9. The next step is to create the Worker class. To do so, extend the IImportExportWorker interface and create a class.

   Here the class is named as ExampleWorker. Override the methods according to the required functionality.

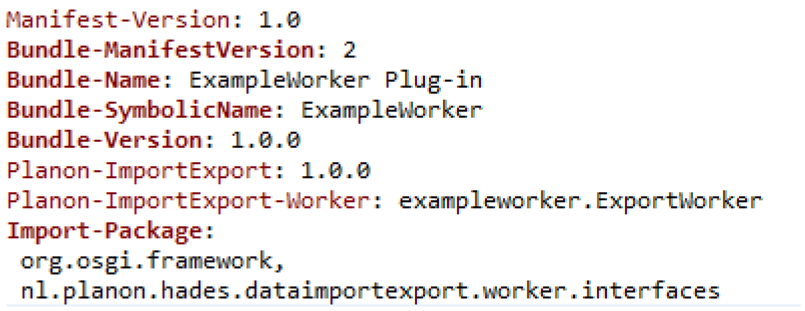
Include the magic header lines in the Manifest file. This is required, because Planon provides the user with a generic Activator and Factory classes for the import/export functionality. This frees the developer from writing his/her own Activator.

In the Manifest file, under **Bundle-RequiredExecutionEnvironment**, include the following lines.

```
Planon-ImportExport: 1.0.0

Planon-ImportExport-Worker: exampleworker.ExportWorker
```

The first line should remain as is. The second line should have the fully qualified name of the worker as header value.

Your manifest.mf should resemble the following sample:



```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: ExampleWorker Plug-in
Bundle-SymbolicName: ExampleWorker
Bundle-Version: 1.0.0
Planon-ImportExport: 1.0.0
Planon-ImportExport-Worker: exampleworker.ExportWorker
Import-Package:
 org.osgi.framework,
 nl.planon.hades.dataimportexport.worker.interfaces
```

10. The next step is to build and deploy the developed code as a bundle. In Eclipse, right-click the developed project, and select Export.

    Choose **Plug-in-development.**

    In the next screen, select your bundle and specify the location where this has to be installed.

    Now your bundle is ready.

    Copy the generated .jar file to the following location: “<Applicationserver>\server\default\bundles\planon”.

    Now you have deployed your bundle.

    All that remains is to configure the worker in Planon ProCenter .

    In **Enterprise Talk**, create a new worker, specify the sequence where it has to execute, and enter the fully qualified name of the Worker developed as plug-in name.

For additional information, see the Java Doc.

Deploying a worker (bundle)

The worker bundles are located in the application server directory: \server\default \bundles. This folder contains two subfolders:

- system
  Containing OSGI framework bundles.
- planon
  Containing specific worker bundles. This is the location where customized workers/worker bundles are to be deployed.

  To deploy a worker/worker bundle, simply place your specific worker/worker bundle (*.jar) in the planon folder. By virtue of the OSGI framework, your workers will immediately be available to Planon ProCenter .

Troubleshooting

The following topic(s) describe(s) specific information that is important to keep into account.

Amount fields in Excel

When you are using Excel as a tool for capturing data (import/export), note that Excel stores amount values differently than may be expected.

Values in Excel sheets are internally stored as XML (right-click any Excel > 7-Zip > open archive ...).

In the XML, values may be stored using a much finer degree of precision that what is displayed

Example 305.65 might stored as 305.64999999998.

Planon cannot alter these values coming from the XML.

When using Excel in Planon (**Enterprise Talk**, **Data Onboarding**, **SDI Configuration**), this may result in a differences between Excel and Planon.

Workaround

- Change the format of all numeric cells in Excel to text.

  Or, if this does not work...
- Prefix all numbers in all cells with apostrophe. An apostrophe before a cell value forces Excel to interpret the value as text. This is mostly useful for values that look like a number or date.

Supported suffix file format

The following table lists the alphabet and their corresponding value that will be appended to the file name as a suffix during the export of documents.

Alphabet	Corresponding Date/Time component	Example
G	Era designator	AD
y	Year	1996, 96
M	Month in year	July, Jul, 07
w	Week in year	27

Alphabet	Corresponding Date/Time component	Example
W	Week in month	2
D	Day in year	189
d	Day in month	10
F	Day of week in month	2
E	Day in week	Tuesday; Tue
a	am/pm marker	pm
H	Hour in a day (0-23)	0
k	Hour in a day (1-24)	24
K	Hour in am/pm (1-11)	0
h	Hour in am/pm (0-12)	12
m	Minutes in hour	30
s	Seconds in minute	55
S	Millisecond	978
z	Time zone (General time zone)	Pacific Standard Time; PST; GMT-08:00
Z	Time zone (RFC 822 time zone)	-0800

The characters that are not supported in Windows OS are \ / : * ? " < > |.
The characters that are not supported in Unix OS are ¬ ! " £ $ % ^ & * ( ) + = { [}] : ; @ ~ # ? < > , | \ ` ' and any spaces.

Unit of measurement transformation

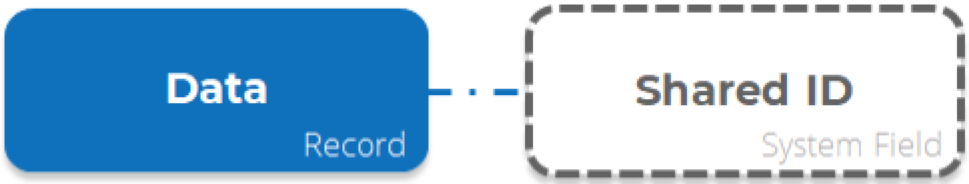
XML header		Database	Result
METER	+	Meters	No data conversion
FEET	+	Feet	No data conversion
FEET	+	Meters	Conversion

XML header		Database	Result
METER	+	Feet	Conversion
-	+	Feet / Meters	No data conversion

Synchronizing records

A **Shared ID** is a unique, mandatory, and insert-only system field.

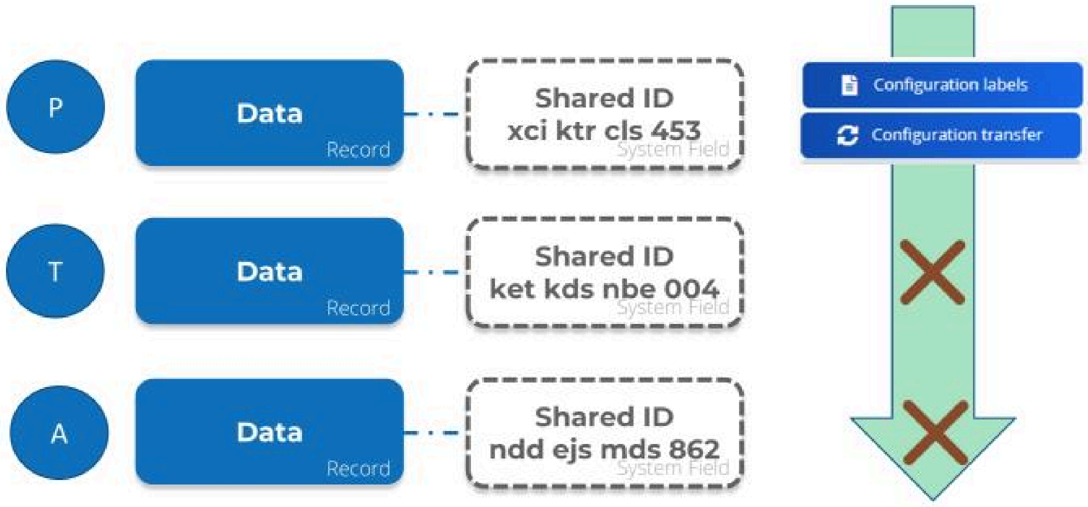
It's purpose is technical: to uniquely identify a record. This is why, while creating a record, the user is not aware of it or needs to worry about it: it is auto-generated during record creation and it cannot be updated later.



Problem

Because of its technical nature, this field is not enabled in GUI and - thus - the user cannot supply a value while creating record via the user interface.

This is problematic when synchronizing data across DTAP environments. If there is no way to synchronize data along with their **Shared IDs**, then the same record in Dev and Test will have different **Shared IDs**, which is functionally incorrect.



This - in turn - is problematic for Configuration Transfer/ Configuration labeling and CAD drawings: it will fail to match the data record with its **Shared ID** in the source and target environments.

Solution

To solve this issue, the **Shared ID** field has been enabled for Enterprise Talk and SDI.

This means that it is now possible to supply the **Shared ID** value while creating a new record using these tools (it needs to be unique, or it will fail).

If you do not provide any value in Enterprise Talk/SDI document or do not include the field in the Enterprise Talk/SDI document, then it will have a unique default value.

How does this work

The **Shared ID** field was introduced in L69. Consequently, when upgrading to L69 (or later) all the relevant records will have this new field, populated with a unique default value.

So, when upgrading DTAP environments separately to L69 this will result in the same records in Dev, Test, Acceptance and Prod having different **Shared IDs** - as already explained.

To solve this issue, we strongly recommend that after the upgrade to L69 is completed to copy the Prod DB to all the lower DTA(P) environments so that for all the existing records the **Shared ID** is in sync across all the DTAP environments.

Technically, it is required only once to copy the Prod DB to lower DTA(P) when you first-time upgrade to L69.

Once all the DTAP environments are in sync you can use the environments as usual.

Going forward, any time if you create a new record in let’s say a Dev environment, then you must import that record along with its **Shared ID** to Test and other environments using Enterprise Talk or SDI. Otherwise, records will not be in sync in the DTAP environments, and this will create issues in Configuration Transfer/Configuration Labeling as discussed.

However, synchronizing the **Shared ID** via Enterprise Talk and SDI only works from L75 onwards. So, any new record can be imported to other environments with its **Shared ID** only from L75 onwards. This means that between L69 to L74 there is no way to sync the **Shared ID** for new records and for that you must upgrade to L75. But existing records will work fine between L69 to L74 - provided you copied the Prod DB to lower DTA(P) environments during the upgrade for L69.

Recap

- When you first-time upgrade to L69 then do not forget to copy the Prod DB to the lower DTA(P) environments.
- If you are below L75, then there is no way to sync the **Shared ID** for the new records.
- From L75 onwards, if you create a new record then you must always first import the data with **Shared ID** to other DTA(P) environments before using Configuration Transfer/Configuration Labeling to avoid the **Shared ID** issues.

As always: first import data, then configuration.

Field descriptions

The following section(s) describe(s) the fields, their purpose and meaning.

Data storage location fields

Field	Description
AWS region	Select the AWS region of your AWS account.
Data storage credentials	Select the account that you want to use. An account can be linked to more than one S3 location.
S3 bucket name	Specify your S3 bucket name, which should already exist.
Root prefix	If you use this field, it will be the root for all data (all files will be relative to this folder). If you leave this field empty, the files will be stored directly in the bucket. You can give a path using slashes (/).

Definition fields

Field	Description
Code	Displays the code of the definition.
Name	Displays the name of the definition.
Use Plaform workers	Indicate if the worker's class originates from an app (new style) or from traditional workers. When set to **No**: default traditional workers are created upon creating an Enterprise Talk Import-export document. When set to **Yes**: app workers will be used.
Data storage location	Select the AWS location. All subsequent paths will be relative to this S3 location. Note that when a path does not yet exist on the S3 location, it will simply be created. For more information, see Using AWS S3 Buckets.

Field	Description
Continue to next document on error	When processing a set of documents, Enterprise Talk needs to know what to do if it encounters an error. If the setting is set to **Yes**, it will continue to the next document or skip documents in case of errors in the current document. If set to **No**, the processing will stop.
Comment	Here, you can add information regarding the definition. You could use it to register version control, add an explanation about its purpose or routine.

Document fields

Field	Description
Code*	Enter a code to identify the document.
Description	Enter a name to identify the import/export document.
Import/export definition*	Select an Import-Export definition that was defined at the **Import-Export definitions** level.
Template for secure document file	Upload the template file that was created in SDI configuration.
Secure document file	Upload the data sheet for onboarding data.
	When you use 'secure documents' as your source documents, you can leave the **Data source** empty. When you do so, both post-processing fields will be disabled.
Data source	Enter the path and file name to the data source to be processed. For import, this is the source file. For export, this is the target file. For example, C:\My documents\Person.xml. It is also possible to read multiple files during import. For more information, see Reading multiple files. The location specified here must be accessible by the application server (and, therefore, by the account that started the application server). For more information on specifying the data source, see Data source.

Field	Description
	In addition to absolute path names (such as C:\My documents\Person.xml), you can also use relative path names. If the **Enterprise Talk root** setting is set, this folder is used to locate the data source. If the **Enterprise Talk root** setting is empty, the location should be relative to the Server\tanuki\appserver\bin folder **Example** ..\..\Talk\person.xmlTalk\person.xml For more information, see the example in Using the Enterprise Talk root folder.
Sequence*	If there are multiple documents available in the list, indicate the order in which the documents should be processed. For example: first import the properties and subsequently the personnel that are linked to these properties.
Document suffix date time format	Enter a valid date/time format that should be appended as suffix to the file name mentioned in the **Data source** field. If the format is incorrect, an error message is displayed. For example, Enter the file name **in Data Source** as C:\MDocs \person.xml. Select **Yes** in the **Export unique file** field. Enter YYYY as suffix format in the **Document suffix date time format** field. Click **Save**. You can preview the unique file name of the exported document in the **Unique file example** field as person_2022.xml. This file format helps you to generate unique file names (in xml, csv and other allowed formats) during export of documents. For more information about the suffix formats, see Appendix.
Export unique file*	Select **Yes** to activate the unique file name feature.

Field	Description
Unique file example	Displays the document file format example as defined in the above fields. For example, person_2022.xml, where person.xml is the file name to which 2022 is appended as the suffix.
Post-process policy for import	Specify the post-process policy for import. This policy only works when using the **Data source** field and not for the **Secure document file** field. Select one of the following options: • **Keep the successfully processed files in the same location** (default): If you select this option, the files, once imported, are not marked nor moved. The imported files remain in the same location. • **Move the successfully processed files to a different location**: If you select this option, the files, once imported, are compressed in a zip file and moved to a new location. • **Delete the successfully processed files**: If you select this option, the files for a document, once imported, are deleted. If any of the files to be deleted is locked, the process skips the current file and tries to delete the next file in the list.
Post-process file location	Specify a folder where the processed file must be saved. In addition to absolute path names (such as C:\My documents\Talk), you can also use relative paths. When the **Enterprise Talk root** setting is set, this folder is used to locate the post-process file location. • This works the same as with the **Data source** field. For more information, see the example in Using the Enterprise Talk root folder. • When using an absolute path name, you cannot use WebDAV locations. However, it is possible to specify a WebDAV folder in the Enterprise Talk root setting.

Field	Description
	For more information about this setting, see File locations (System Settings).
Post-process policy for failed import	Specify the post-process policy for *failed* import. This policy allows you to determine what to do with failed files. • This policy only works when using the **Data source** field and not for the **Secure document file** field. • This policy is not applicable for SDI/Data onboarding definitions. Select one of the following options: • **Keep the processed files in the same location**: If you select this option, the failed files remain in the same location. • **Move the processed files to a different location** (default): If you select this option, the failed files are compressed in a zip file and moved to a new location. • **Delete the processed files**: If you select this option, the failed files are deleted. If any of the files to be deleted is locked, the process skips it and tries to delete the next file in the list.
Post-process file location for failed import	Specify a folder were the failed file must be saved. **Example** You can specify a relative path or an absolute path or an Amazon S3 bucket location - if data storage is configured.
Reference date	**Import** When a date is entered, all life cycles and reference date aware BOs will be added (BeginDate) with this date (see Specifying the reference date). Consequently, if this field is set, the begin dates in the input file will be ignored. If the field is empty, the begin date is taken from the input file (see Importing on a reference date). **Export** When a date is entered, life cycles and reference date aware BOs active on this date are exported

Field	Description
	(see Specifying the reference date). When the field is empty, all life cycles and reference date aware BOs are exported (see Exporting with or without a reference date).
Fetch size	Specify the number of items to split up the document in chunks to prevent a transaction time out. A chunk can be a number of main elements or files. Per Enterprise Talk document, the specified number of items will be processed as separate transactions within the same run. • Default value: 1000 • Minimum value: 100 • Maximum value: 10,000 If the main elements contain several subelements, you should take this into account and adjust the fetch size value accordingly.
Enable	Here, you can determine whether your document should be included in the import/export run. If you click **Yes**, it will be included in the run. Clicking **No** will exclude it from the run. This feature is useful when using a definition with multiple documents. It enables you to test/run a single document (by disabling others) or to exclude a single document (by disabling it).
Auto-approve confirmation answer?	Indicate whether to automatically accept or reject confirmation messages that occur when running an import. By default, Enterprise Talk *answers* all confirmations with **No**. If Approved messages has been configured, this takes precedence over **Auto-approve confirmation answer?**.

Import Business object definition fields

Field	Description
Business object*	Select the business object into which you want to import data from the data source. The data source is specified at the **Documents** level.

Field	Description
	In some business objects, its code does not uniquely identify the business object. The BO can be identified only when the code is in combination with another field. For example, for Space BO, the combination of **Code** and **Property** makes the space unique for that Property. This is because the space with the same code can be present in a different property. Hence, for both Code and Property the **Part of search key** option in the **Definition Details** level should be marked **Yes**. Similarly, while importing an Asset, two business objects, Asset and Location Assignment have to be configured. The Location assignment BO should specify the Asset BO as a ‘SUPER’ in the configuration.
Document*	Select the document to which the selected business object belongs.
If record occurs once*	Indicate how the import must “act” if a record exists once. This field only applies to importing business objects. **Create/Update** Updates the record. **Skip** Skips this record without modifying it. **Stop** Stops the import.
If record exists twice or more*	Indicate how the import must “act” if a record exists twice or more. This field must have a value when you want to import data from an XML file. **Create/Update** Updates all records. **Skip** Skips this record without modifying it. **Stop** Stops the import.
If no records exists	Indicate how the import must “act” if a record does not exist. This field must have a value when you want to import data from an XML file.

Field	Description
	Create/Update Creates the record. **Skip** Skips this record without modifying it. **Stop** Stops the import.
Marking current state	You can select a regular or a Free field to mark the current state of import. This field indicates the records that were not imported this time from the last imported set. The Free field should be of type **Mark** for it to be available for the import.
Import/export marked	Allows you to mark imported records in Planon ProCenter so that you can easily know which records were not imported during the current run. Select **Yes**, to mark the records during import.
Mark field	Allows you to select a Free field to mark the record during the import. For example, **Free string 8.** Marking of records is not supported when used with **Database import-export worker**. The Free fields become available, only when the **In use** field is marked **Yes** in Field definer . Note that the field type (of the Free field) should be of type **Mark**. For more information, refer to Field definer . If **In use** is marked **No**, an error message is displayed. Refer to the import/server log to know which field is not imported.
Mark value	Enter the value that should be inserted into the Free field for marking the imported record. That is, the Free field selected in Mark field will be marked with the value specified in this field. For example, if you specify the value as Mark records for SAP, the Free string 8 (selected in Mark field) will acquire the value as Mark records for SAP after the import.

Field	Description
XML node name*	Specify the name to be used as XML tag. For valid XML, the tag name must comply with the rules for XML element names: Names can contain 0-9,a-z,A-Z,_ (numbers, lowercase letters, CAPITAL letters, and an underscore (_)). Names cannot start with a number or punctuation character. Names must start with a letter. Names cannot start with the letters xml (or XML, or Xml, and so on). Names must not contain spaces. For more information on XML element names, see:http://www.w3schools.com/xml/xml_elements.asp

Export Business object definition fields

Field	Description
Business Object*	Select the business object from which you want to export data to a data source. The data source is specified at the **Import-export documents** level.
Document*	Select the document to which the selected business object belongs.
Main business object (Y/N)*	Indicate whether the selected business object is the main business object. The main business object is the starting point for the export. A main business object will always be exported. A business object not marked as main will only be exported if it is included in the export definition of another business object.
XML node name*	Specify the name to be used as XML tag. For valid XML, the tag name must comply with the rules for XML element names: Names can contain 0-9,a-z,A-Z,_ numbers, lowercase letters, CAPITAL letters, and an underscore (_).

Field	Description
	Names cannot start with a number or punctuation character. Names must start with a letter. Names cannot start with the letters xml (or XML, or Xml, and so on). Names must not contain spaces. For more information on XML element names, see: http://www.w3schools.com/xml/xml_elements.asp
Filter	Specify the filter conditions to be applied to the export. The filter only applies to main business object. If no filter is specified, all data will be exported.
Sequence	If there are multiple main business object definitions available in the list, indicate the sequence order in which the business object definitions should be processed. This field applies to export only; for import, the order in the XML file is leading.
Mark records on an external table	If this field is set to **Yes**, records are created in the separate export mark list step (sublevel of Talk export BO) and an extra marker worker is added (MarkExportWorker). The worker is removed when marking is switched off. This mark worker (nl.planon.morpheus.pnworkers.server.planonreaderworker.MarkExportW should be the last worker and should not be removed. The worker is only for exporting.
Export policy	Allows you to determine if you want to export all business objects or export only the non-marked instances that match the filter criteria. **Skip if marked as export**: select this option to export only the non-marked BOs. This prevents the export of the previously exported records and saves a lot of export time. **Always**: select this option to export all records matching the filter criteria.

Field	Description
	Export Policy is independent of the setting on the **Marking of records for export**.
Mark record for export	Allows you to mark the records for export. (A Free field is used for marking.) Select **Yes**, to mark the record for export. If you select **No**, the record will not be marked.
Mark field	Allows you to select a Free field to mark the record with text value during the export. For example, **Free string 14.** The Free fields become available, only when the **In use** field is marked **Yes** and the Field type as **Mark** in Field definer . For more information, refer to Field definer .
Mark date - time	Allows you to select a Free field to mark the record with date-time value during the export. If **In use** is marked **No**, an error message is displayed. For more information, refer to the import/server log to know which field is not imported.
Mark value	Enter the value that should be inserted into the Free field for marking the exported record. That is, the Free field selected in **Mark field** will be marked with the value specified in this field. For example, if you specify the value as **Mark records for SAP,** the **Free string 14** (selected in **Mark field**) will acquire the value as **Mark records for SAP** after the export.

Import-Export worker fields

On the Business object definition > Import-export worker step, a number of settings allow you to configure workers.

Field	Description
Enable logging	Specify whether you want to receive a log of all business objects that were imported successfully.

Field	Description
Import-export document	Displays the document for which you are configuring workers.
Class name	Class that implements the worker. If the field Use Platform workers is **Yes**, only classes originating from an app are shown. If the field is **No**, only traditional style worker classes are shown.
Sequence	Indicate the order in which this worker should operate in the list of workers. The order of processing is determined by the sequence number given here. The process starts with the lowest number.
App module name	The name of the module in the app from which the selected class originates.
App name	The name of the app from which the selected class originates.
Partner identifier	The Partner identifier of the app from which the selected class originates.
Settings tab	
Settings	The settings that apply to the worker (JSON format). The user has to define the values of the various settings in this field. When the app is upgraded (introducing new or changing settings), this field is not automatically changed. When this happens, you must manually update this field.
Settings example tab	
Settings example	A calculated field with example of the settings, in JSON format. This is intended to help define the correct settings in the **Settings** field. When the app is upgraded (introducing new or changing settings), this field is updated according to the new settings.
Settings schema tab	
Settings schema	A calculated field with the type/format and constraints that apply to the settings, in JSON format.

Field	Description
	When the app is upgraded (introducing new or changing settings), this field is updated according to the new settings.

Field definitions & default values to fields

Field	Description
Field definition*	Select the specific field for which you want to import or export a value.
Mandatory in Talk? *	Specify whether this field is mandatory for the talk import.
Import-export business object*	Select the business object for which you want to import or export data.
Part of search key?*	Indicate whether the field specified in the **Field definition** field is (part of) the search criteria for looking up the correct business object. You can select multiple fields for identifying a business object. If a field is marked as search key and a business object is found, this field will not be updated. Consequently, read-only or system fields can be used as search key.
XML node name*	Specify the name of the tag in the XML file corresponding with the field whose value you want to import or export. A value is only required for field references. For an association, the XML node name will be derived from the business object definition.
DefaultValueXML	A read only field that displays the default value.
Default field value policy	Select one of the import policies form the list: • **Never use default**: selected by default. • **Use default when the field does not exist**: When you select this option, the value specified in the Default value field is used, if the field does not exist in the import file. • **Use default when the field is empty or does not exist**: When

Field	Description
	you select this option, the value specified in the **Default value** field is used, if the field is empty or if it does not exist in the import file.
Default field value	Select a default field value for the import. This value is used to import field values which are not present in the import document but are required as part of the imported data. used to mention a default value of a field chosen for an Import business object definition.

Log / Log details fields

Field	Description
Application log type	Displays the type of log, which is either: • Error • Warning • Information
Source	Displays information that identifies the import/ export.
Log message	Displays information that will help you understand what went wrong during the import/export.

Field as sub

Field	Description
Field definition	Select the field referencing the business object for which you want to import data.
Part of search key*	Indicate whether the selected field is (part of) the search criteria used for identifying the correct business object. To uniquely identify a business object, it is possible to specify multiple fields. If a field is marked as search key and a business object is found, this field will not be updated. Consequently, read-only or system fields can be used as search key.

Field	Description
Import-export business object definition*	Select the import-export business object definition for which you want to import data or from which you want to export data.
Referenced import-export definition*	Select the field referring to the business object into which you want to import data.

Field as super

Field	Description
Field definition	Select the field referencing the business object for which you want to import data.
Part of search key*	Indicate whether the selected field is (part of) the search criteria used for identifying the correct business object. To uniquely identify a business object, it is possible to specify multiple fields. If a field is marked as search key and a business object is found, this field will not be updated. Consequently, read-only or system fields can be used as search key.
Import-export business object definition*	Select the import-export business object into which you want to import data or from which you want to export data.
Referenced import-export definition*	Select the field referring to the business object into which you want to import data. Example Import all properties and the persons linked those properties.

Transforming data fields

Field	Description
Code	Enter a code or name for the worker.
Enable logging	Specify whether you want to receive a log of all business objects that were imported successfully. Especially when failures are reported, a log with all successfully imported

Field	Description
	business objects will provide a means of checking and assuring the actual imports. The list of successfully imported business objects is shown on the **Definition details** selection level > **Business object log** step. This step is only available from the **Business object definitions** selection level: Select a log entry > click the **Definition details** selection level. For more information on logging, see Logging.
Class name	Specify the path and name of the worker/ worker bundle or select a worker from the list of registered Enterprise Talk workers.
Sequence	Indicate the order in which this worker should operate in the list of workers. The order of processing is determined by the sequence number given here. The process starts with the lowest number.
Additional parameters	Here, you can specify all required parameters needed by the worker. For standard workers, the needed parameters can be found in the chapter describing using standard workers. Use a new line to separate parameters, in some cases a comma is also possible. When defining the worker's additional parameters, it is possible to comment out a parameter using the hashtag (#) before the parameter, except for the XSLT worker.
Description	Provide a description about the purpose of the worker (optional).

Worker parameters

The following sections describe the parameters that can be used in the corresponding workers.

You can specify parameters in multiple lines by separating them by new lines.

DBReaderWorker parameters

Parameters	Description
Query.<main/sub>.<node>.Sql	SQL (select) statement to be executed on the specified database in order to export the right data. If the configured SQL does not contain valid XML, the processing is aborted. Whenever JAVA special characters are used in the SQL statement these should be escaped by a backslash (\).
UnitOfMeasurement	Unit of measurement used in the source. This value is used for the header element. M/F M:Meters F:Feet Default value: **M**
EmptyFields	An XML element is created for an empty column value in the source database when the value is set to **True**. Default value: **True** **True**: the XML gets an element without a value (Empty or NULL). **False**: the XML will not get an element for the field.
database-name	The JNDI name mentioned in the DS File. For example, database-name some-mssql-DS. JNDI name in the ds.xml file should end with 'DS'.

Parameters	Description
database-name	The JNDI name of the data source you would like to export from, as mentioned in the *…\Server \wildfly-*\standalone\configuration\standalone-full.xml* file. The **database -name** param value is present as a part of file name present in the folder <install-location>\ Server\<jboss-XX>\server\default\deploy. The data source should be named <name>DS, for example Worker-DS.

CSVReaderWorker parameters

Parameters	Description
mainElementName	The main element name that will be used in the XML generated by the CSVReaderWorker. If this value is not specified, the value will be taken from the Node name of the BO definition if there is only one, or by default it is **element**.
header	Specify **Yes** or **No** if the CSV to import contains a header row with the names of the column in it. By default, the value is set to **Yes** if this attribute is not specified.
columns	The name given to the XML element tags of the columns if no header is available. If a header is available, it specifies the order or a subset of columns that must be imported. If not specified and the CSV has a header row, the order from the header is used and all columns are imported. If not specified and the header is absent, the default column names are column-1, column-2, etc.
delimiter	The valid values for the delimiter are: • ; or semi-colon or semicolon or semi • tab or \\t • space or \\s • , or comma • any literal symbol

Parameters	Description
	Semi-colon is the default value of the delimiter.
textQualifier	Valid values for a text qualifier are: • ' or apos or apostrophe • " or quote or quot • any literal symbol
trim	Set the value to **No**. This indicates that the leading and trailing whitespace around the CSV data should not be trimmed.
fileEncoding	If not specified, utf-8 is used if the file encoding is detected automatically. Otherwise, the file encoding specified is used.

CSV WriterWorker parameters

Parameter	Description
mainElementName	By default, the value of the main element will be taken from the BO definition, if there is only one BO. Otherwise, it is **element**, unless a name for this parameter is specified in the parameter section.
header	Specify a **Yes** or **No** for the CSV header. By default, the value is set to **Yes**. If this attribute is not specified, a header is created.
columns	Names of the columns will be taken from the XML element tags. This setting will determine the order or you can specify the subset of the XML element tags. If this parameter is not entered, the order from XML is used and all elements are exported. The column names are case-sensitive.
delimiter	The valid values for a delimiter are: ;semi-colon or semi tab or \t space or \s

Parameter	Description
	or comma and any literal symbol. Semi-colon is the default value of the delimiter.
textQualifier	Valid values for a text qualifier are: ’ or apostrophe or apos and ” or quote or quot.
ifFileExists	Possible values: • replace or overwrite • cancel or stop • new or create: This will create a new file by using the file name and append it with a timestamp.
writeutf8Bom	This BOM is written only if the file encoding is UTF-8. This indicates that the UTF-8 order mark should be written.
generate_file_per_bo	Specify ***true*** or ***false***. Default is false. If set to true, a CSV file will be generated for each row. The name of each CSV will be equal to the name specified in the document data source with an extra underscore [_] and a sequence number added to the name.

ExcelReaderWorker

Parameters	Description
tabSheet	Excel sheet name.
mainElementName	Business object name.
columns	Columns to process separated by ;.
rowHeader	Row that contains the header. - **Default value**: ‘1’
rowStartData	Row number from where the data starts.
numberOfRows	Number of rows to process. **Default value**: process all the rows available in the excel sheet.

Parameters	Description
columnForStopDef	Column name to stop reading of rows. If this column is empty the reader will stop reading. **Default value**: ‘A’
nrOfEmptyRowsToStop	Number of empty rows to stop reading of rows. **Default value**: ‘5’
unitOfMeasurement	M/F(Meters/Feet) **Default value**: 'M'
referenceDateCell	Cell reference containing the reference date. **Default value**: System date. When a reference date is filled in the document, this reference date will be used.
referenceDate	Reference date **Default value**: System date. Example: 2015-01-31
referenceDateElementName	XML node name containing the reference date from the referenceDateCell.

FileXMLWriterWorker parameters

Parameters	Description
generate_file_per_bo	Specify ***true*** or ***false***. Default is false. If set to true then an XML file will be generated for each row. The name of each XML will be equal to the name specified in the document data source with an extra underscore [_] and a sequence number added to the name.

element_per_line	Specify ***true*** or ***false***. Default is false. If set to true then each line in a multiline field will be enclosed in seperate XML tags.
	Example - false:
	`<property>`
	`<comment>this`
	`is`

Parameters	Description

	```
	    a
	    multiline
	  </comment>
	</property>
	```
	 Example - true:
	```
	  <property>
	   <comment>
	   <TEXTLINE>this</TEXTLINE>
	   <TEXTLINE>is</TEXTLINE>
	   <TEXTLINE>a</TEXTLINE>
	   <TEXTLINE>multiline</TEXTLINE>
	  </comment>
	</property>
	```

XSL Transformation parameters

Parameters	Description
xslfile	The file containing the XSL (optional). You should specify only xsl file path and not the complete xsl content. For example, xslfile=C:\temp\budget.xsl. In addition to absolute path names, you can also use relative path names. The location should be relative to the ...\Server\tanuki \appserver\bin folder.

XSLT Transformation Worker parameters

Parameter	Description
in	The input file (optional). If not set, the value of the source field specified on the Import Document will be used. In addition to absolute path names, you can also use relative path names. The location should be relative to the ...\Server\tanuki\appserver\bin folder.
out	The output file. This is a mandatory field. In addition to absolute path names, you can also use relative path names. The location should be relative to the ...\Server\tanuki\appserver\bin folder.
format	The format of the XSLT result (optional). The default is “xml”; other valid values are "html", "xhtml" and "text".
xsl	A string containing the XSL to apply (optional). It is not possible to use a comma (,) or an equal sign (=) in the xsl parameter. If you need to use these characters, you can use the xslfile parameter.
xslfile	The file containing the XSL (optional). You should specify either xsl or xslfile. In addition to absolute path names, you can also use relative path names. The location should be relative to the ...\Server\tanuki\appserver\bin folder.

Standard Worker Properties

The following overview lists all standard workers with their paths and respective in & out parameter types.

Name	Path	IN	OUT
Import Workers			
FetchFilesWorker	nl.planon.morpheus.pnworkers.server.-xmlworkers.FetchFilesWorker	File set	File
	This worker only works when using the **Data source** field and not when using the **Secure document file** field.		
XMLReaderWorker	nl.planon.morpheus.pnworkers.server.-xmlworkers.XMLReaderWorker	XML file	XML string
XMLToPOJOWorker	nl.planon.morpheus.pnworkers.server.-xmlworkers.XMLToPOJOWorker	XML string	POJO object
PlanonDataWriter-Worker	nl.planon.morpheus.pnworkers.server.-planonwriterworker.PlanonDataWriterWorker	POJO object	DB
CSVReaderWorker	nl.planon.morpheus.pnworkers.server.-csvworkers.CSVReaderWorker	CVS file	XML string
ExcelReaderWorker	nl.planon.morpheus.pnworkers.server.-excelworker.ExcelReaderWorker	Excel file	XML string
DBReaderWorker	nl.planon.morpheus.pnworkers.server.-dbreaderworker.DBReaderWorker	DB	XML string
Export Workers			
PlanonDataReader-Worker	nl.planon.morpheus.pnworkers.server.-planonreaderworker.PlanonDataReaderWorker	DB	POJO object
FileXMLWriterWorker	nl.planon.morpheus.pnworkers.server.-xmlworkers.FileXMLWriterWorker	POJO object	XML file
CSVWriterWorker	nl.planon.morpheus.pnworkers.server.-csvworkers.CSVWriterWorker	XML string	CSV file
POJOToXMLStrin-gWorker	nl.planon.morpheus.pnworkers.server.-xmlworkers.POJOToXMLStringWorker	POJO object	XML string

Name	Path	IN	OUT
MarkerExportWorker	nl.planon.morpheus.pnworkers.server.-planonreaderworker.MarkExportWorker		

Bi-directional Workers

Name	Path	IN	OUT
XMLChunkXSLTTransformationWorker	nl.planon.morpheus.pnworkers.server.xsltworkers.XMLChunkXSLTTransformationWorker	XML string	XML string
FileXSLTTransformationWorker	nl.planon.morpheus.pnworkers.server.-xsltworkers.FileXSLTTransformationWorker	XML file	XML file

Index