Interface Toolbox for Human Resources (PX-XX-TL)

HELP.PAXX

Release 4.6C

Copyright

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax
	Tip

Contents

Interface Toolbox for Human Resources (PX-XX-TL)

Interface Toolbox for Human Resources (PX-XX-TL)

Purpose

In the SAP System, you can use the following methods to transfer data from the application components in *Human Resources* (HR) to a third-party system, or vice versa:

- Interface Toolbox
- Application Link Enabling (ALE [Ext.])
- Business Application Programming Interface (BAPI [Ext.])

A comparison of the technology [Page 10] shows which method is most suitable to perform the task.

You use the Interface Toolbox to retrieve data from *Human Resources* for further processing in a third-party system. The third-party system can be a payroll system used outside of the SAP System. This can be useful if, for example, you run gross payroll in your enterprise and the net payroll takes place in a third-party system.

Alternatively, you can use the Toolbox to import results based on personnel numbers from the third-party system and use them for payroll in the SAP System.

You use the interface to

- Evaluate data from *Human Resources* in a third-party system
- Run payroll partially or completely in a third-party system

The data is retrieved from the *Personnel Administration* (PA-PA), *Payroll* (PY), and *Time Management* components, or from all three.

Access

You access the Toolbox using transaction PU12.

In the *Time Management* component (PT), you call the Interface Toolbox from the menu. Choose *Time Management* → *Administration* → *Environment* → *Third-party payroll.*.

In the *Payroll* component (PY), you also call the Interface Toolbox from the menu. In the *SAP Menu*, choose *Human resources* → *Payroll* → *Tools* → *Maintenance tools* → *Interface Toolbox*.

Features

Selected examples show which scenarios can be performed using the Toolbox:

- Exporting master data to a third-party system [Page 12]
- Exporting master data and payroll results to a third-party system [Page 14]
- Gross payroll in an SAP System and net payroll in a third-party system [Page 16]

The Toolbox enables you to create an interface as well as carry out routine work. The following processes and associated functions are important for using the Interface Toolbox:

- For the data export

  Setting up the interface for exporting with the Toolbox [Page 18]

Interface Toolbox for Human Resources (PX-XX-TL)

Data export with the Toolbox [Page 20]

- For the data import

  Importing wage types with the Toolbox [Page 183]

  For detailed information on the required activities, see the individual sections.

Technology for Interface Scenarios

Task	Method
Export HR master data **(infotypes) from the SAP System to a third-party system**	Interface Toolbox • Transport HR master data from all possible infotypes (also customer infotypes) • Convert selected data to file format for third-party system • Example: Export master data to a third-party system [Page 12] **ALE [Ext.]** (Application Link Enabling) • The standard system contains an ALE business process for some infotypes • You can insert additional infotypes (see note in OSS note system) Data is exported in the SAP format only (no conversion)
Import HR master data **(infotypes) from the third-party system to the SAP System**	ALE / **BAPI [Ext.]s (Business Application Programming Interface)** • Batch input technology [Ext.]
Export payroll results **(infotypes) from the SAP System to a third-party system**	Interface Toolbox • Select all possible tables in Payroll • Additional options for processing wage types (for example, consideration of retroactive accounting) • Example: Export master data and payroll results to a third-party system [Page 14] Combination of Toolbox and ALE • The Toolbox fills the IDocs • Distribute the complete IDocs using ALE • IDocs exist in the SAP interface for North America - they can be used by customers (in IDoc administration, these are the IDoc types that begin with 'HROT')

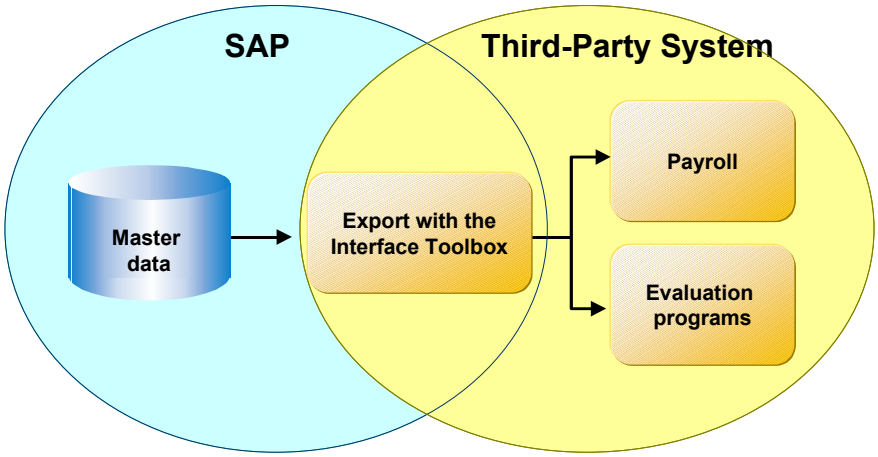
Import wage types **from a third-party system to the SAP System**	Process **(See also: Importing Wage Types with the Interface Toolbox [Page 183])** **1. Transfer the wage types to interface tables with** – Transfer by customer program – A BAPI (Object: BUS7023 Manager for External Payroll; Method: InsertOutsourcer) 2. Start payroll to transfer the wage type to the payroll result
Export time evaluation results **(infotypes) from the SAP System to a third-party system**	Interface Toolbox **Selection of Time Management tables that are relevant for payroll (tables: ZL, C1, ALP)**
Import time evaluation results **from the third-party system to the SAP System**	**Human Resources and external applications [Ext.]**
Change validation **(Transport changed data to third-party system)**	**Interface Toolbox** **(See also:** Change Validation [Page 56]**)** • Change validation is possible for individual fields or for complete infotype records • Method: Import all data and compare it with old data (check performance). ALE • Change validation only possible on infotype record level (export complete infotype record) • Method: Change display technology (system recognizes changes to the infotype and only exports these if a change takes place)

Example 1: Export Master Data to a Third-Party System

Task

You want to export HR master data from the SAP System to a third-party system to be used as a basis for

- Evaluations
- Payroll



This application example is simple to effect with the Interface Toolbox.

Only master data is taken from the SAP System. No information is transferred from the third-party system to the SAP System.

Prerequisites

You use the *Personnel Administration* application component (PA-PA) in your enterprise.

Method

Step	in the SAP System	In the Third-Party System
1. Export HR master data (infotypes) from the SAP System to a third-party system	With the Toolbox • Set up the interface for the export • Run the data export	The master data and payroll results from the SAP System are in the third-party system.

2. You start payroll or payroll reporting programs in the third-party system.		The payroll and reporting results are available in the third-party system.

Example 2: Export Master Data and Payroll Results to Third-Party System

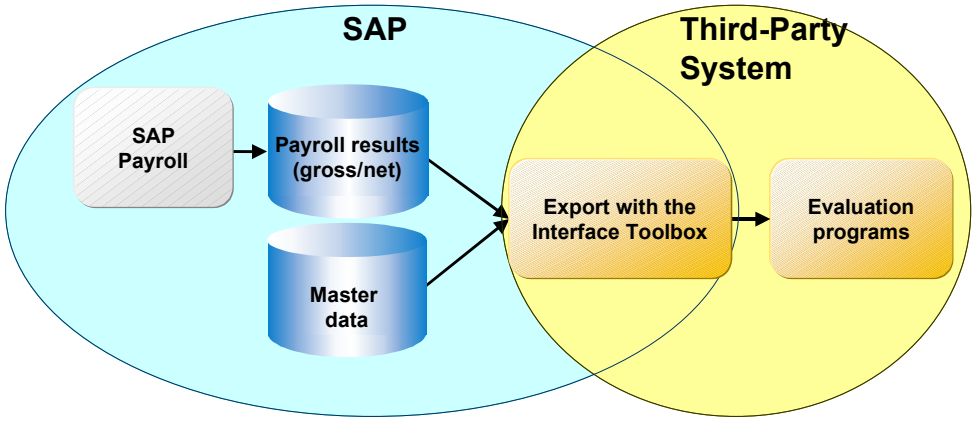
Task

You want to use HR master data and payroll results (gross or net) from the SAP System to a third-party system as a basis for

- Evaluations
- Payroll

You can also transfer time results to the third-party system instead of the payroll results.

No information is transferred from the third-party system to the SAP System.



Prerequisites

You use the following application components in your enterprise:

- *Personnel Administration* (PA-PA)
- *Time Management* (PT)
- *Payroll* (PY)

You have set up gross payroll in the SAP System.

Method

Step	in the SAP System	In the Third-Party System
1. You run payroll (gross or net) in the SAP System.	The payroll results are available after a successful payroll run.	

2. You export the master data (infotypes) and the payroll results to the third-party system.	With the Toolbox • Set up the interface for the export • Run the data export	The master data and payroll results from the SAP System are in the third-party system.
3. Start the reporting programs in the third-party system.		The reporting results are available in the third-party system.

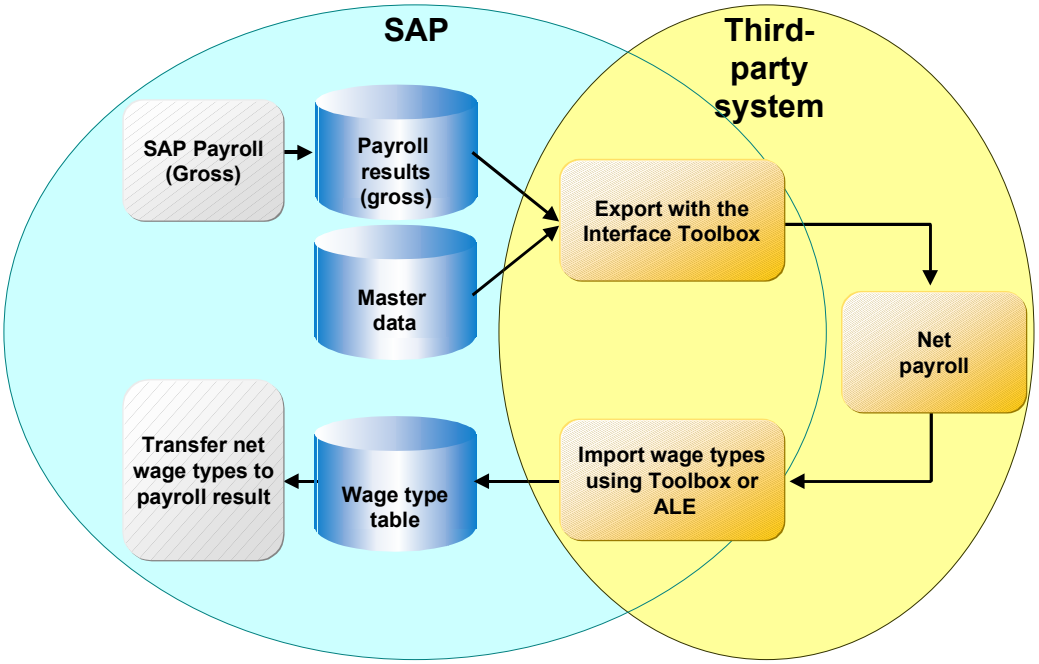
Example 3: Gross Payroll in SAP System, Net Payroll in Third-Party System

Task

You want to use master data and payroll results (gross) from the SAP System as the basis for further payroll in a third-party system. You also want to transfer wage types (net) from the third-party system to the SAP System.

In the SAP System, you require information on the net wage types if you

- Run posting to Accounting with the *Human Resources* component (HR)
- Print remuneration statements or checks in the SAP System
- Start programs for reporting in the SAP System
- Collect all wage type information in the payroll result for the SAP System



Prerequisites

You use the following application components in your enterprise:

- *Personnel Administration* (PA-PA)
- *Payroll* (PY)

You have set up gross payroll in the SAP System.

Method

Step	in the SAP System	In the Third-Party System
1. You run payroll (gross) in the SAP System.	The payroll results (gross) are available after a successful payroll run.	
2. You export the master data (infotypes) and the payroll results (gross) to the third-party system.	With the Toolbox • Set up the interface for the export • Run the data export	The master data and payroll results from the SAP System are in the third-party system.
3. Start payroll (net) in the third-party system.		The payroll results (net) are available after a successful payroll run.
4. You import the payroll wage types (net) from the third-party system to the SAP System.	With the Toolbox • Import the wage types from the third-party system to the SAP System	
5. Start payroll in the SAP System to transfer the payroll wage types (net) to the payroll results.	In the SAP System, the wage type information is integrated with the payroll results.	

Setting Up the Interface for Export with the Toolbox

Purpose

You use this process to

- Set up the interface for exporting with the Toolbox
- Export the selected data to the third-party system using the Interface Toolbox.

Prerequisites

You use the following application components in your enterprise:

- *Personnel Administration* (PA-PA)
- *Time Management* (PT)
- *Payroll* (PY)

Process flow

The process is split into several steps:

2. You use the Toolbox to create an interface format [Page 21]. In this step, you select the data and convert [Page 39] the data to be exported.

   The interface format includes the following functions:

   - Create objects
   - Delete objects
   - Data definition for cluster objects
   - Table entries
   - Convert field objects
   - Restrictions for field objects
   - Attributes

3. You determine which data is only to be exported if it has changed.

   You need only perform this step if only changed data is to be exported.

   You use the change validation functions [Page 56] to determine the data for the export. These functions include:

   - Create objects
   - Delimit objects
   - Validating single fields
   - Key fields
   - Relations

- Wage types

3. You use the data entered in the interface format and in change validation to generate the export program [Page 115].

   You use the functions in the file layout [Page 129] to set up the required file format for the data to be exported. These functions include:

   - Create
   - Delete
   - Move
   - Insert
   - Attributes
   - Blocks
   - Structures
   - Fields

4. You use the specifications for the file layout to generate the conversion program [Page 156].

Result

You have set up an interface for the export. You can start the generated export program to run the data export and convert the data to the required file format.

Data Export with the Toolbox

Purpose

You use this process to retrieve data from the *Human Resources* application components (HR) for Payroll or for the payroll reporting programs for use in a third-party system.

Prerequisites

You use the following application components in your enterprise:

- *Personnel Administration* (PA-PA)
- *Time Management* (PT)
- *Payroll* (PY)

You have used the Interface Toolbox to set up an interface for the export (see application example 1 [Page 12]) or you are using an interface from the standard system.

Process flow

The process is split into several steps:

1. With the Toolbox, start the export program [Page 117].
2. You use the Toolbox to download the export files from the TemSe file to your PC or to an application server (Downloading the Export Files [Page 128]).
3. If the system administrator for this system is in agreement, you can transfer the data to the third-party system.

Result

The third-party system contains data from the SAP System. You can use this data for Payroll or for the payroll evaluation programs.

Interface Format

Definition

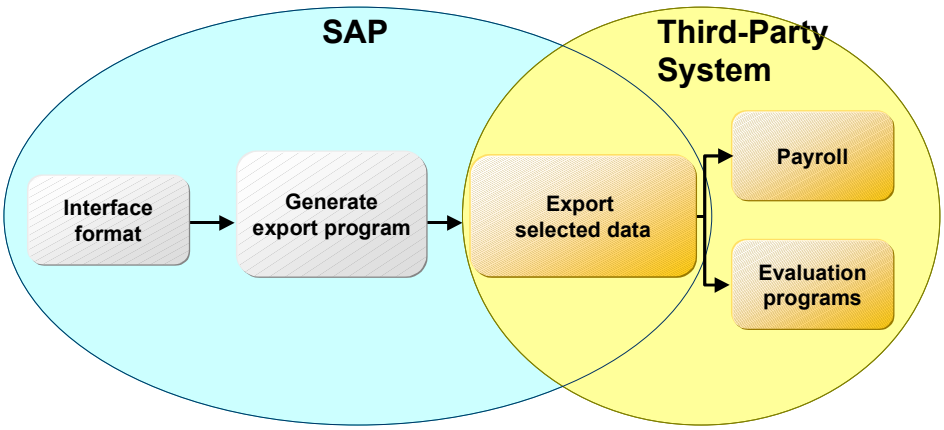
The interface format contains the objects that you want to export from the *Human Resources* application component (HR) to a third-party system.

Use

The interface format is the basis for data export with the Interface Toolbox. You use the interface format to determine which *HR* data you want to select, and also how this selection should take place. When you define an interface format, you can access all employee data for the following application components:

- *Personnel Administration* (PA-PA)
- *Time Management* (PT)
- *Payroll* (PY)

Using your defined interface format as a basis, the Interface Toolbox will ultimately generate a program using the *Advanced Business Application Programming* language (ABAP [Ext.]). You use this program to export the data. The system processes the data quickly and effectively during the export.



Structure

The interface format is a hierarchical sequence of objects.

Objects and their hierarchy

Object	Hierarchy classification	Example
Database objects [Page 25]	1	MDTA for HR master data PCL2 for payroll
Cluster objects [Page 26]	2	RX, RD, ... for payroll results B2 for time evaluation results

Interface Format

Table objects [Page 27]	3	WPBP for work center/basic pay, P0002 for personal data
Field objects [Page 29]	4	PERSK for employee group ANZHL for number

Task

To save the personnel number data, the *interface format* uses the objects in the *Human Resources* component (HR). You use the interface format to select a data subset that meets your requirements.

You can choose to select only the infotypes that you require for processing in a third-party system.

You can also select particular fields from the fields in an infotype.

The interface format and objects exactly match the definition of the data to be exported later with the export program. For performance reasons, the set of export data should be arranged in the best possible way.

Characteristic

Each object in the Interface Toolbox has specific characteristics. The characteristics determine how the object is processed and are assigned as follows:

- Entered explicitly by the user

  You assign the *Conversion* characteristic to a field object.

- Implicitly by the Interface Toolbox

  The Interface Toolbox automatically assigns the *Length* characteristic to the field object.

- By the SAP System

  The SAP System assigns the *Permissibility per infotype* characteristic to the field object.

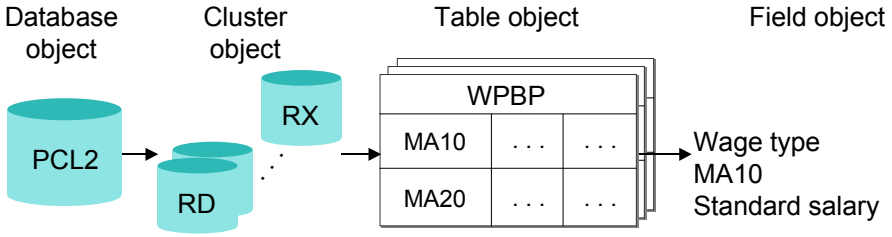
Create Objects

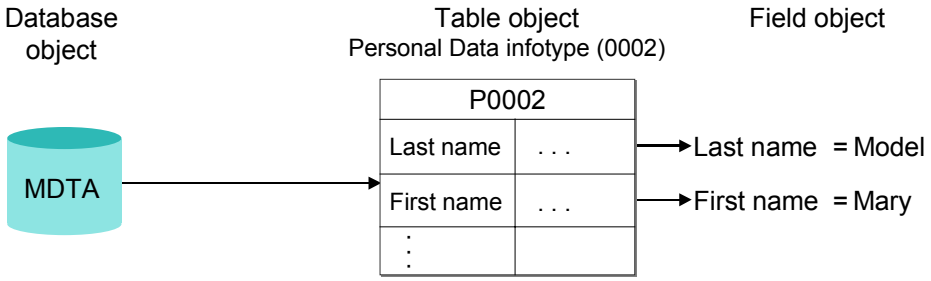
Use

The *Create* function is permitted for all objects.

- Database objects [Page 25]
- Cluster objects [Page 26]
- Table objects [Page 27]
- Field objects [Page 29]

You use the function to create an object in the tree structure and the accompanying characteristics.





Different dialogs are used depending on the object, and you must enter information for the object to be created.

To use the *Create* function efficiently and to create the data consistently, you must always create the parts of the objects that are subordinate in the hierarchy.

You want to create a database object with the type Master Data (MDTA). To do this, select the required infotypes.

You also want to create a database object with the type *Export-Import File* (PCL2). To do this, you must select the required cluster object (for example, RX) and then select the required objects from the table objects belonging to the cluster (for example, WPBP).

The field object is an exception. If you create a table object, the system will automatically transfer all assigned field objects to the interface format.

Activities

You create an interface format [Page 30].

Database Object

Definition

Subset of tables from a database.

You can use the Interface Toolbox to select the following database objects:

- Set of all infotypes (MDTA) with the type *Master Data*
- Payroll and Time Management data (PCL2) with the type *Import/Export file*

Use

You use the database object to create an interface format. When you create an interface format, note the following:

- Database objects with the type *Master Data* (MDTA) are followed by table objects. Infotypes belong to the table objects.

  If you create a database object with this type, you can assign required infotypes for the subsequent export.

  The table object (infotype) automatically includes all field objects belonging to the relevant infotype.

- If database objects have the type *Import/Export File* (PCL2), then the subordinate hierarchy level consists of cluster objects.

  When you create a database object with this characteristic, you must enter the name of the database table on which the interface format is based (for example, PCL2). The Interface Toolbox will then check whether the specified database table exists in the SAP System.

  You must also maintain the following objects which are lower in the hierarchy:

  – Cluster object

  – Table object

    The table object automatically includes all field objects belonging to the table.

    You can also create an additional database object at a later point.

You then run

- The dialog for a table object for master data
- The dialog for a cluster object for an import/export file

Cluster Object

Definition

Each database object, for example, PCL1 or PCL2, with the type *Import/Export file* consists of related areas. These areas are known as *clusters*, for example. RX, RD. Cluster objects are dependent on the superordinate database object.

You can select the following cluster objects for database object PCL2:

- Cluster RD
- Cluster RX
- Cluster B2

If the database object has the type *Master Data*, then there is **no** cluster object.

Use

If you assign a database object with the type *Import/Export file* to the interface format, you must select the cluster in which the payroll or time evaluation data is found.

When the interface format is processing cluster data, it uses the existing import macros [Ext.] in the SAP System. If you have changed the table objects for the data import in the import macro (for example, if you have deleted or inserted a table object), these changes are automatically transferred to the Interface Toolbox.

The Toolbox checks that the import macro is actually reading the data from the superordinate database object and the specified cluster.

The system checks that the data definition include [Page 53] and the import macro are syntactically correct.

If you create a new cluster object or change the characteristics of an existing cluster object, you must enter the following information in the system:

- Name of the cluster (for example, B2)
- Name of the data definition include (for example, RPC2RDD0)

  The data definition include contains the definition of the data defined by the import macro.
- Import macro (for example, RP-IMP-C2-RD)

  The import macro determines the set of all possible subsequent table objects.

You then run the dialog for the table objects.

Table Object

Definition

Table from *Human Resources* (HR).

The system differentiates between the following types:

- Section of a cluster object

  The table object, for example, WPBP, is a section of a cluster object, for example, RX, which belongs in turn to the database object, for example, PCL2.

- Infotype

  The table object, for example, P0002, is an infotype whose superordinate database object (MDTA) is the set of all infotypes.

Characteristic

The only available characteristics for the table object are those automatically assigned by the SAP System.

Characteristics that you have assigned using the *Table Entries* function are an exception. For more information, see Table Entries [Page 37].

The system derives the characteristics from the data definition include (for example, RPC2RDD0) for the superordinate cluster object (for example, RD).

For more information, see the section on data definition of cluster data with the data definition include [Page 53].

For table objects with the *Set of All Infotypes* as the superordinate database object (MDTA), the system derives the characteristics from the ABAP Dictionary (DDIC) [Ext.].

Use

If you assign a database object with the type *Import/Export file* to the interface format, you must select the cluster containing the payroll or time evaluation data.

When you create a table object, the system displays a list of all internal tables and field strings. There following types of lists are available:

- List of all internal tables or field strings, if the table object belongs to a cluster object.
- List of all infotypes, if the table object belongs to a database object with the type *Master Data*.

From this list, you can select the table objects for data export. The Interface Toolbox automatically inserts all accompanying field objects for the selected table object.

Field Object

Definition

The field object, for example, BUKRS for company code, is a field in the superordinate table object, for example, the *Organizational Assignment* infotype (P0001). The system first assigns characteristics to the field object based on the ABAP Dictionary [Ext.]. You can change these later by converting the field object.

Characteristic

- Type (for example, CHAR)
- Length (for example, 4)
- Content (for example, 0001)

Use

Each table object contains a particular number of field objects. A list of all fields belonging to the corresponding table object is displayed. The Interface Toolbox creates the list using the structure assigned to the table object from the *ABAP Dictionary*. From this list you can select all the field objects whose data you want to export.

Creating an Interface Format

Use

You want to create an interface format with the name **Z000**. The interface format **Z000** should contain database objects:

- *Personal data* (P0002):
- *Basic pay* (P0008)

You want to use **all** field objects from both database objects.

Procedure

1. Choose the *Configuration* tab index.
2. In the *Object to be processed* group box, choose *Interface Format*.
3. In the *Interface format* field, enter the name of your new interface format `Z000`.
4. Choose *Create*.

   You access the *Create Interface Format* dialog box.
5. In the *Country grouping* field, enter the country indicator and, in the *Description of new interface format* field, enter a text.

   You access the *Create database object* dialog box.
6. To create a database object with the type *Master Data*, choose the *Master Data*.

   To create an interface format with the type *Import/Export file*, you should be familiar with the import macro for cluster data [Ext.] and cluster data definition using a data definition include [Page 53].

   You access the *Create master data* dialog box.
7. Select the *Personal Data (P0002)* and *Basic Pay (P0008)* infotypes.

   The *Export program* dialog box is displayed.
8. Enter data as required.

   If you choose *Suggest values,* the system proposes names for the export program and includes.
9. Choose *Continue.*
10. Choose *Edit* → *Expand*.

    The system displays all levels of the object tree with the individual field objects for the infotypes.
11. Save your entries.

    You access the *Create object catalog entry* dialog box.

12. Enter the development class and choose *Save.*

Result

You have created an interface format with the name **Z000**.

You can use the individual interface format functions to edit the generated interface format. For example, you can insert a field object in an interface format [Page 32].

Inserting a Field Object

Use

You can use the *Create* function to insert field objects in a table object and consequently, alter the sequence of the field objects within a table object. The system inserts the selected field objects after the current cursor position.

Prerequisites

You have created an interface format.

Procedure

1. Choose the *Configuration* tab index.
2. In the *Object to be processed* group box, choose *Interface Format*.
3. In the *Interface format* field, enter the name of your interface format.
4. Choose *Change.*

   The system displays the **tree structure**.
5. Choose *Edit* → *Expand*.

   The system displays all **table objects** with the accompanying **field objects**.
6. In the selected table object, position your cursor on the field object after which you want to insert one or more field objects.
7. Choose *Create*.

   You access the *Add Fields from <name>* dialog box.
8. Select the field object(s) you want to add to the table object.

   The system inserts the selected field objects **after** the current cursor position.

   If you select a field object from the list that is already assigned to the relevant table object, then this field object will be ignored and not transferred.
9. Save your entries.

Result

You have added one or more field objects to the table object.

Delete Objects

Use

The *Delete* function is permitted for all objects.

- Database objects [Page 25]
- Cluster objects [Page 26]
- Table objects [Page 27]
- Field objects [Page 29]

You use the function to remove an object from the tree structure.

If you delete an object from the tree structure, you also remove all objects below it in the hierarchy.

You also want to create a database object with the type *Export-Import File* (PCL2).

The accompanying cluster objects, table objects, and field objects are also deleted.

Integration

If you delete objects from the tree structure of the Interface Toolbox , the change validation [Page 56] objects derived from these objects will also be deleted.

This means that the relationship between the object in the interface format and in *change validation* is always **consistent**.

For example, if you delete the field object (LGART) from the table object (RT), this field object will also be deleted from the change validation tree structure.

Activities

Delete an object [Page 34].

Deleting Objects

Prerequisites

You have created an interface format.

You are in the tree structure [Page 32] and all table objects are displayed with all field objects.

Procedure

1. Select the objects to be deleted using *Select.*

   When you select an object, the Interface Toolbox includes all dependent objects in the object hierarchy. This means that all objects subordinate to the object to be deleted will also be selected.

2. Choose *Delete*.

   The system deletes **all** selected objects including those subordinate to the object to be deleted.

3. Save your entries.

Result

You have deleted objects from the tree structure.

Data Definition for Cluster Objects

Use

The *Data Definition* function is only valid for cluster objects [Page 26].

This function allows you to change the name of a data definition include [Page 53] for a cluster object.

If you create a new database object or cluster object for the interface format, the Interface Toolbox requires the following information:

- Name of the cluster
- Name of the data definition include
- Name of import macro

If you change the name of the data definition include you should ensure that the new data definition include contains at least all the data definitions from the old include.

Activities

Either change the name of a data definition include [Page 36] or display the include.

Changing or Displaying the Data Definition for a Cluster Object

Prerequisites

You have created an interface format.

You are in the tree structure [Page 32] and all table objects are displayed with all field objects.

Procedure

1. Position the cursor on the required cluster object.
2. Choose *Data definition*.

   You access the *Data Definition for DB <name>, Cluster <name>* dialog box.
3. You can now change the name of the data definition include or display the include.

   The dialog box also gives you an overview of the assignment of internal table names or field string names to the corresponding structures in the data dictionary.
4. If you have changed the name of the data definition include, choose *Save*.

Result

You have changed the name of your data definition include and you have displayed an overview of the assignments to cluster objects.

Table Entries

Use

The *Table entries* function only applies to table objects [Page 27].

WPBP (*Work center/Basic pay*)

Basic pay infotype (P0008)

This function allows you to use the Interface Toolbox to select records for data export from the input data stream. You can specify whether you want to export all table entries or just a selection of entries.

Activities

Select table entries [Page 38] for a table object.

Selecting Table Entries

Prerequisites

You have created an interface format.

You are in the tree structure [Page 32] and all table objects are displayed with all field objects.

Procedure

1. Place the cursor on the selected table object.
2. Choose *Table entries*.

   You access the *Table entries for <name>* dialog box.
3. Choose an entry for
   a. For an infotype, choose
      – All records valid at the start date of the current period
      – All records valid at the end date of the current period
      – All records valid on at least one day in the current period
      – All existing records, including those outside of the current period
   b. A cluster
      – First entry
      – Last entry
      – All entries in the current period
      – All entries
4. Save your entries.

Result

The result of this selection is only apparent after the data export, for example, in the export program log, or directly in the output file.

Conversion for Field Objects

Use

The *Conversion* function only applies to field objects [Page 29].

The Interface Toolbox allows you to modify the export file to meet the requirements of the third-party system, as well as allowing you to transfer the data in the SAP System to the third-party system. The *conversion* then modifies the data.

During the conversion, the system replaces the old content (value) of the object with the new content (value).

- The *Gender Key* field object (P0002-GESCH) has the content (value) **2,** which should be replaced by the content (value) **1** during the export.
- The *Form of Address Key* field object (P0002-ANRED) has the content (value) **1,** which should be replaced by a **text** (for example, Dear Sir).
- The names of the wage types in the SAP System (for example, /101) can be replaced by the names of the wage types used in the third-party system (for example, GROSS).

Features

The system differentiates between three conversion types for field objects:

- Direct value [Page 40]
- Table value [Page 43]
- User exit [Page 46]

The Toolbox also includes additional conversion options in the file layout [Page 129] . You can use these options in addition to or instead of the conversion options in the interface format.

Activities

Select the conversion for a field object [Page 48].

Constant Conversion Type

Use

The *Constant* conversion type causes the Interface Toolbox to replace the old value of a field object by a new value during the data export. You can enter the necessary default values for the conversion of the field objects in the *<name> (old)* and *<name>(new)* fields of the *Direct Value for <name>* dialog box.

You want to replace the old value of the ANSSA field object, which is contained in the *Addresses* infotype *(P0006),* with a new value.

Enter the following in the *Constant for "Address Record Type"* dialog box.

Field: ANSSA (old)	Field: ANSSA (new)
1	Main address
2	Temporary address
*	Holiday address

Features

Replace Constant Generically

Use the * (asterisk) character to replace values generically. In your selected field object, enter a * (asterisk) in the... (old) field as the first and only character. Enter the new constant in the ... (new) field.

If, when processing the data records, the Toolbox finds the character * (asterisk) for generic replacement, the system replaces all values without a suitable entry in the ...(old) field with the value from the ...(new) field for the generic lines.

See also:

Example: Replacing a Constant Generically [Page 42]

Length of the Field Object for the Constant Conversion Type

You can use the *Length* function to **redefine** the field length of a field object. Normally the Toolbox uses the length in the ABAP dictionary as the length of the field object. The new field length can be either longer than, equal to, or shorter than the length in the ABAP dictionary.

Reference for the Constant Conversion Type

You can use the *Reference* function to refer to an existing, suitable conversion. You can define a reference to an existing conversion program without creating a new conversion program. If you modify the fields in the *Constant Value for <name>* dialog box then the Toolbox will also integrate these changes in the referenced fields.

If suitable references for the field object to be converted already exist in the SAP System, then they will be shown in the *Reference Conversion Type* dialog box and they can be used again

Your payroll result contains the table objects A and B. The LGART field object is contained in table A and table B.

You define the constant conversion type for the LGART field object from table object A.

You only define one **reference** from the LGART field object in table object B to the LGART field object in table A.

If you modify the *Constant* conversion type in one of the two fields (A-LGART, B-LGART), then the Toolbox will automatically integrate these changes in the referenced fields.

Transfer of Entries for Conversion

Once you have made all the entries for the conversion of a field object, choose *Transfer*.

Example: Replacing a Constant Generically

Purpose

You want to replace the selected values of the KOSTL field object *(cost center)* with new values during the data export.

Process flow

Perform the conversion of direct values in two steps:

1. Enter the following values in the *KOSTL (old)* and *KOSTL (new)* fields in the *Constant for Cost Center* dialog box.

Field: KOSTL (old)	Field: KOSTL (new)
1111	Z1111
2222	Z222
*	ZZZZ

2. When the Interface Toolbox exports data, selected values for the *Cost Center (KOSTL)* field are contained in the input data stream records. The system replaces these selected input values for the *Old cost center (KOSTL (old))* field by the values you have entered in the *New cost center (KOSTL (new))* field.

Values for the KOSTL field in the input data stream for the export	Content of the KOSTL field after the data export
1111	Z1111
2222	Z2222
3333	ZZZZZ

Table Value Conversion Type

Use

The *Table Value* conversion type operates in a similar way to the *Constant* conversion type where the Interface Toolbox replaces the old value of the field object with a new value during the export process. However it differs from the constant conversion type because the new value contained in a field, for example, TITEL for title, comes from a database table [Page 44], for example, T535N for name supplements. The new value is a value stored in the SAP System and is not determined by a constant.

To use the table value conversion type, you need to know the table names of the required database tables (for example, table T535N for name supplements) and fields (for example, the TITEL field for title).

Key Fields for the Table Value Conversion Type

The content of the key fields defines the access path for the selected field, whose content is to be used as the replacement.

If you enter the name of the database table and field name and choose *Enter*, then the Toolbox automatically enters the default values from the SAP System in the key fields for this table.

The access path for the key fields is a default value in the SAP System. Check and, if necessary, correct this access path so that the correct field object can be used for the table value conversion.

In addition, the SAP System allows you to use system fields that are available in the *System Fields in ABAP Programs (SYST)* table. These are the fields that begin with **SY-**, for example, SY-LANGU for the language key in the SAP logon procedure, and SY-DATUM for the current date in the system: You can use these system fields to define the access path to a field object.

Database Table

Database tables used when converting table values are all tables in the SAP System that exist in the ABAP Dictionary (DDIC). This does **not** just include the database objects used by the Interface Toolbox for the interface format.

Example: Replacing a Table Value

You can use the *table value* conversion type is to replace values that represent a code with values that represent a text and which already exist in a field in the database table.

You want to replace the value in the *Form of Address (ANRED)* field object in the *Personal Data (P0002)* infotype with the value from the *Form of Address text (ATEXT)* in the *Forms of Address (T522T)* database table.

You are in *Conversion* → *Table value* for the *Form of Address (ANRED)* field object from the *Personal Data (P0002)* table object.

Enter **`T522T`** in the *Table* field and **`ATEXT`** in the *Field Name* field in the *Table Values for "Form of Address Key"* dialog box. When you choose *Enter,* the key fields P0002-SPRSL and P002-ANRED are displayed, provided you have selected both fields for the export.

You end the example with the *Transfer* function.

User Exit Conversion Type

Use

If the functions offered by the *Constant* and *Table Value* conversion types do not meet your requirements, the Interface Toolbox allows you to use your own, user-defined conversion program. The *User Exit* conversion type creates the link to the customer conversion program.

Prerequisites

The following conditions must be met before you can define the field object conversion for the *User Exit* conversion type in the interface format:

- You have generated a user-defined program for the conversion of a field object.

  The name of the user-defined program matches the program name in the *Program* field.

- The routine specified in the *Form Routine* field is specified in your user-defined program.

  The SAP System checks that the specified form routine exists in your user program.

- The Interface Toolbox provides you with a default value for the length of the field object to be converted. You can modify this length to suit your requirements at any time.

The *User Exit* conversion is carried out for each table line in the selected table object (for example, WPBP) in the input data stream. Define the form routine in your user-defined program with two parameters:

Parameters	Meaning
Parameter 1	Current line of the table object in which the field object is contained (input parameter)
Parameter 2	Result of conversion (return value) The return value is interpreted as a character field type.

See also:

Example: User Exit with Form Routine [Page 47]

Example: User Exit with Form Routine

Basic Situation

You use a customer *User Exit* with the program name ZUSEREXI and the form routine with the name KONVERT.

You wish to convert the value in the *Title* (TITEL) field object in the *Personal Data* infotype (P0002) to the text from the *Title* (TTOUT) field in the *Name Supplements* (T535N) table. The title ING (for engineer) should not be used.

Procedure

The customer program for the user exit (including the form routine) could be as follows:

```
     FORM KONVERT TITEL USING    VALUE(P0002) STRUCTURE P0002
                         CHANGING VALUE(RESULT).
          IF P0002-TITEL <> `ING.`.
             SELECT SINGLE * FROM T535N
                                  WHERE ART = `T`
                                  AND TITEL = P0002-TITEL.
         IF SY-SUBRC = 0.
                RESULT = T535N-TTOUT.
   ELSE.
                CLEAR RESULT.
   ENDIF.
ENDFORM.
```

Selecting the Conversion for A Field Object

Prerequisites

You have created an interface format.

You are in the tree structure [Page 32] and all table objects are displayed with all field objects.

Procedure

5. Place the cursor on the selected table object.
6. Choose *Table entries*.

   You access the *Table entries for <name>* dialog box.
7. Place the cursor on the selected field object and choose *Conversion*.

   The *Conversion type* dialog box appears.
8. Choose one of the following conversion types:
   - *Constant*

     You access the *Direct Value for <name>* dialog box.

     a) Enter data as required.

     b) Choose *Transfer.*
   - *Table value*

     You access the *Table Value for <name>* dialog box.

     a) Enter data as required.

     b) Choose *Transfer.*
   - *User Exit*

     You access the *User Exit for <name>* dialog box.

     a) Enter the required data.

Field	Content
Program	Program name of the user-defined user exit
Form routine	Name of form routine
Conversion length	Length of converted field object

     b) Choose *Continue.*
9. Save your entries.

Result

You have assigned a conversion type to the selected field object.

Restrictions for Field Objects

Use

The *Restrictions* function only applies to field objects [Page 29].

The *Address record type* field (ANSSA) in the *Addresses* infotype (P0006).

The *Pay scale type* field (TAFAR) from table WPBP (Work center/Basic pay).

You can use this function to select records from the input data stream to be exported by the Interface Toolbox. The selection criterion is the field object with the predetermined restrictions. You can use your default values to define the restrictions referring to the content of the selected field object. Only the data records whose field object matches that in the *Restrictions* function will be exported from the input data stream.

You only want to select the main address from the ANSSA (address type) field in the *Addresses* infotype (P0006). You do not need the other address types.

To do this, enter **1** for address in the *Restrictions for "P0006-ANSSA"* dialog box and copy the entry.

The Interface Toolbox only exports the input records for the data stream that have 1 (main address) in the ANSSA field. Data records with other addresses are not exported.

Restriction types for field objects

- Special restrictions for wage types

  This type of restriction is only valid for wage types from wage type tables.

  For more information on wage types, see Wage Type Processing in the Interface Toolbox [Page 86].

- General restrictions

  This type of restriction is valid for field objects with the type CHAR and NUM.

Activities

Select the restrictions for a field object [Page 51].

Creating Restrictions for a Field Object

Prerequisites

You have created an interface format.

You are in the tree structure [Page 32] and all table objects are displayed with all field objects.

Procedure

1. Place the cursor on the selected field object.
2. Choose *Restrictions.*

   You access the *Restrictions for <name>* dialog box.
3. If you have already defined restrictions for the field object and want to insert more restrictions, choose *New line*.
4. Enter the restrictions.

   Several entries for a field object are linked implicitly with the logical operation **OR.**
5. Choose *Transfer.*
6. Save your entries.

Result

You have created a restriction for the selected field object.

Attributes in the Interface Format

Use

You use the *Attributes* function to define the processing parameters for the interface format. These are used by system when the data is exported. They determine how the interface format will be used in the SAP System.

Features

The attributes can be changed during or after the processing of the interface format.

For the process control:

- Interface without payroll
- Check control record
- No retroactive accounting
- Valuate wage types indirectly
- Layout conversion directly after export
- New change validation [Page 60]

For storing the interface results:

- Do not save to cluster IF
- Save current results only

Activities

You use the *Attributes* function to assign the attributes at any time during the editing of the interface format.

You must assign the attributes to the interface format **before** you generate the export program.

If you assign the attributes to the interface format later, you **must** regenerate the export program.

Data Definition Include

Definition

The data definition include contains the definitions of the internal tables and field strings used in the import macro [Ext.]. Like the import macro, it is a further characteristic of the cluster object [Page 26]. There is just one include for each cluster object, and it contains the definition of the data.

Use

The Interface Toolbox must recognize the structure of the cluster to be able to read cluster objects in the SAP System. The data definition include contains this structure definition. The Toolbox uses the data definition include to read the data from a cluster.

Activities

The Interface Toolbox enables you to generate a data definition include automatically [Page 54] using the specifications for this program.

Creating an Include Automatically or Using an Existing Include

Procedure

1. Choose the *Configuration* tab index.
2. In the *Object to be processed* group box, choose *Interface Format*.
3. In the *Interface format* field, enter the name of your new interface format.
4. Choose *Create*.

   You access the *Create Interface Format* dialog box.
5. In the *Country grouping* field, enter the country indicator and, in the *Description of new interface format* field, enter a text.

   You access the *Create database object* dialog box.
6. Choose *Payroll results/Time data.*

   You access the *Create Payroll Results/Time Data* dialog box.
7. In the *File Data* data group, enter the following fields:

Import/Export Table	Name of import/export file (for example, PCL2)
Name of the cluster	Name of the cluster (for example, RU)

8. Choose *Continue.*

   The system suggests values for the automatic generation of data definition include.

   For example, RP-IMP-C2-RD is displayed in the *Name of import macro* field.

   For example, *RPCALCU0* is displayed in the *Name of program* field.
9. Choose one of the following steps:

   - **Create a data definition include automatically**

     a) So that the system will generate the data definition include, enter a new name for the include (for example, ZPC2RU00) in the *Include to be created* field of the *Data definition include* group. If necessary, change the names of the import macro and program.

     The program name for the cluster containing payroll results (Rx) is the program name of the payroll driver that you use in Payroll.

     - RPCALCx0; where x is the country indicator, for example, RPCALCD0
     - HxxCALC0; where xx is the country indicator, for example, AR for Argentina

     The program name for the time evaluation cluster B2 is the international name RPTIME00 (*Time Evaluation)*.

b) To generate the include, choose *Continue.*

You access the *Create Object Catalog* dialog box.

c) In the subsequent dialog, enter the corresponding object catalog entry.

You access the *Add Tables/Field Strings* dialog box.

– **Use an existing include**

You have already created a data definition include for another interface format for the same cluster object, and want to use this include.

a) To use this data definition include, choose *Existing include* in the *Data definition include* data group. In the *Name of include* field, enter the name.

b) Choose *Continue.*

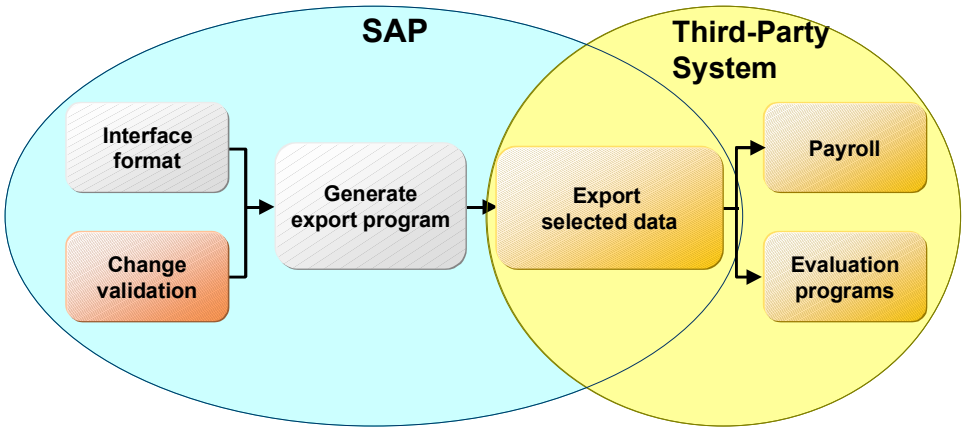
The *Insert Tables/Field Strings* dialog box is displayed.

10. Select the required objects.
11. Choose *Continue.*
12. Save your entries.

Result

You have created your data definition include automatically or used an existing data definition include.

Change Validation



Use

Most of the data exported to a third-party system using the Interface Toolbox **does not** change in every payroll period [Ext.]. For example, an employee's personal details (name, first name, personnel number, and so on) rarely differ from one period to the next.

If the Interface Toolbox transfers all data for a payroll period for every data export, the following disadvantages arise:

- Increased redundancy of exported data (larger volume of data)
- Increased transfer load during the data export
- Increased processing

  The receiver **must** process all transferred information as it is not possible to determine in advance whether or not the information has changed during the payroll periods.
- High probability of errors made by the Interface Toolbox during data export and during the processing of the information by the receiver

It is therefore a good idea to only export data that has changed since the last data export. The third-party system can use the data that has not changed since the last data export again since this data already meets your requirements. Only changes to the data must be transferred to the third-party system for further processing.

The *Change Validation* function allows you to avoid exporting unchanged data.

Prerequisites

You have created an Interface Format and can only call the objects assigned to the Interface Format for *change validation*. The objects for *change validation* include:

- Table objects [Page 27] (internal tables, field strings, and infotypes)
- Field objects [Page 29]

For more information on the interface format and objects, see Interface Format [Page 21].

Features

To deal with this problem, the Interface Toolbox has a *change validation* function with a wide range of functions.

- Create objects [Page 67]
- Delimit objects [Page 69]
- Single field validation [Page 72]
- Key fields [Page 75]
- Relations [Page 78]
- Wage types [Page 81]
- User-defined change validation [Page 82]

There are special rules for the change validation of wage types from the corresponding wage type tables. For more information, see Wage Type Processing in the Interface Toolbox [Page 86] .

See also:

Comparison Period for Change Validation [Page 58]

Comparison Period for Change Validation

Definition

The comparison period is a payroll period used in change validation to determine which data has changed since the last export.

Use

The *Change Validation* function allows the Interface Toolbox to determine which data has changed from one period to another. This data is then transferred to the export file and this determines the volume of data to be exported.

If retroactive accounting is first excluded from individual payroll periods and from multiple exports, then it is easy to determine which data in a payroll period will be used for comparison. A comparison is always made between the data for the current export and the data for the last export, since it identifies the changes that have occurred in the data to be exported.

Data from the March payroll period is compared with the data from the February payroll period when the data is exported. Differences in the March payroll period when compared to the February payroll period are exported.

If data is exported for an employee for the first time, there is no comparison period for payroll. If this is the case, the system exports the data for the existing payroll period.

The determination of payroll periods for retroactive accounting and multiple exports is explained in the following sections.

See also:

Setting the Comparison Period for Multiple Export [Page 59]

Setting the Comparison Period for Retroactive Accounting [Page 60]

Setting the Comparison Period for Multiple Export

Use

If you run the export program several times for the same payroll period, then this is known as **multiple export.** Multiple export can be split into two types:

- Repeat
- Additional export

To identify the changes in the dataset for multiple export, the system must know the payroll period. When you define the payroll period, you should note the following:

The selection screen for the export program [Page 117] contains a *Repeat run* parameter in the *Export options* group box.

- **Select the checkbox for a repeat run.**

  The last data export for this payroll period did not take place for the export program.

  The change validation function uses the **last but one** data export from the selected payroll period to compare the data. An export must have taken place previously, otherwise the comparison will be made with the most recent comparison period.

- **Do not select the checkbox for an additional export.**

  If another export run takes place during the selected payroll period, the export program assumes that the third-party system has received the data from the last export.

  The change validation function uses the **last** data export from the selected payroll period to compare the data. An export must have taken place previously, otherwise the most recent period will be used for the comparison.

See also:

Setting the Comparison Period for Retroactive Accounting [Page 60]

Setting the Comparison Period for Retroactive Accounting

Use

There are two ways of setting the comparison period for retroactive accounting:

- First method

  The third-party system receives the data from the original period as well as from the retroactive accounting period.

  - If the *New change validation* attribute is flagged, the Toolbox uses the new variant for the first method [Page 61]
  - If the *New change validation* attribute is **not** flagged, the Toolbox uses the old variant for the first method [Page 63]

- Second method [Page 65]

  The third-party system only receives data from the original period.

Determination of Comparison Period Using First Method if *New change validation* Attribute Is Flagged

Prerequisites

The Toolbox uses this variant if you

1. Have flagged the *New change validation* attribute [Page 52] in the interface format
2. Have **not** flagged the *Only original periods* export option for the export.

   It is assumed that the third-party system has received the data from the retroactive accounting periods as well as from the original periods.

Use

We recommend that you use this variant because the disadvantages of the old variant [Page 63] do not apply. The Toolbox now uses the new variant as the standard for a new interface format.

Advantages

- Each change is only sent once per payroll period. If subsequent exports follow the first export, in contrast to the old variant, these changes are not sent.
- The Toolbox also sends retroactive changes in all cases.

  For the first export after a change that covers several payroll periods, the changes are sent for each payroll period affected by the change.

In retroactive accounting for payroll, the preceding payroll periods are defined using the rule that the **last original period** is always used as the comparison period.

Rule

The system always determines the most recent payroll period in which the for-period view [Ext.] matches the payroll period. If this does not exist, the last exported payroll period is used. The payroll period can be a previous retroactive period for the current export if it exists or the last payroll period of the past export.

These rules are described in an example.

The payroll periods listed in the table are applicable for an employee.

					New	New	New	Old	Old	Old
Sequential no.	**For-period**	**In-period**	**Contents field 1**	**Contents field 2**	**Preceding period**	**Contents field 1 sent**	**Contents field 2 sent**	**Preceding period**	**Contents field 1 sent**	**Contents field 2 sent**

Determination of Comparison Period Using First Method if New change validation Attribute Is Flagged

1	01 2000	01 2000	A	X	-	A	X	-	A	X
2	02 2000	02 2000	B	X	1	B	#	1	B	#
3	01 2000	02 2000	A	Y	1	#	Y	1	#	Y
4	02 2000	02 2000	B	Y	2	#	Y	3	B	#

Determination of Comparison Period Using First Method if *New change validation* Attribute Is Not Flagged

Prerequisites

The Toolbox uses this variant if you

1. Have **not** flagged the *New change validation* attribute [Page 52] in the interface format
2. Have **not** flagged the *Only original periods* export option for the export.

   It is assumed that the third-party system has received the data from the retroactive accounting periods as well as from the original periods.

Use

The specification of the precedent for the payroll period in change validation has been converted. For compatibility reasons, SAP supports this variant.

Disadvantages

- If a payroll period is exported several times, changes are also transferred several times.
- In certain situations, the Toolbox does not export retroactive changes.

In retroactive accounting for payroll, the preceding payroll periods are defined using the following rules:

Rule 1

For the **first** retroactive accounting period in payroll, the preceding period is the most recent payroll period in the in-period view [Ext.] that matches the current payroll period in the for-period view [Ext.]. If this payroll period does not exist, then the system does not call up a preceding period for the data comparison.

Rule 2

For the **following** retroactive accounting periods and the original period for payroll, the system always uses the preceding retroactive accounting periods.

These rules are described in an example.

The payroll periods listed in the table are applicable for an employee.

Sequential no.	For-period	In-period	Preceding period	Explanation
00001	01 2000	01 2000	none	First export
00002	02 2000	02 2000	00001	
00003	03 2000	03 2000	00002	
00004	04 2000	04 2000	00003	

Determination of Comparison Period Using First Method if New change validation Attribute Is Not Flagged

00005	02 2000	05 2000	**00002**	Preceding period according to rule 1
00006	03 2000	05 2000	**00005**	Preceding period according to rule 2
00007	04 2000	05 2000	**00006**	Preceding period according to rule 2
00008	05 2000	05 2000	**00007**	Preceding period according to rule 2

Second Method for Setting the Comparison Period

Use

It is assumed that the third-party system has not received the data from the retroactive accounting periods. The third-party system only receives data from the original period.

In retroactive accounting for payroll, the preceding payroll periods are defined using the rule that the **last original period** is always used as the comparison period.

Rule

The system always uses the **last original period** as the comparison period.

Application and Procedure for this rule

On the selection screen for the export program [Page 117] , select the *Only Original Periods* radio button in the *Export Options* group box. The export program then uses the **original payroll periods** to check for changes.

Result

The retroactive periods in payroll are suppressed for the data export.

The comparison period for change validation is always an original period.

The payroll periods listed in the table are applicable for an employee.

Only the original periods should be called up for the data export. The third-party system does not contain the data from the retroactive accounting periods.

Sequential no.	For-period	In-period	Preceding period	Period type
00001	01 1996	01 1996	No preceding period	O
00002	02 1996	02 1996	00001	O
00003	03 1996	03 1996	00002	O
00004	04 1996	04 1996	00003	O
00005	02 1996	05 1996	No export	R
00006	03 1996	05 1996	No export	R
00007	04 1996	05 1996	No export	R
00008	05 1996	05 1996	00004	O

Legend

O Original period

R Retroactive accounting period

Second Method for Setting the Comparison Period

Create Objects

Use

The *Create* function for change validation only applies to table objects [Page 27].

This function checks a table objects and accompanying field objects assigned to the interface format for changes to the dataset between two exports. The system compares an object from one data export with the same object from the previous data export and checks for changes. The table object and all field objects lower in the hierarchy are included in data validation.

For example, the *Personal Data* infotype *(P0002)* includes all field objects that the infotype has in the selected interface format.

If you do not use any other c*hange validation* functions, the system performs change validation for the table objects using the accompanying field objects. If change validation identifies **at least one change** in an object, then the export program will **transfer** the **complete table object** with data to the third-party system.

Activities

Create table objects for change validation [Page 68].

Creating Objects

Prerequisites

You have created an interface format.

Procedure

10. Choose the *Configuration* tab index.
11. In the *Object to be processed* group box, choose *Interface Format*.
12. In the *Interface format* field, enter the name of your interface format.
13. Choose *Change Validation.*
14. Choose *Display / Change*.

    The system changes to the change mode.
15. Choose *Create*.

    You access the *Field Strings/Tables* dialog box.
16. Select the table object for *change validation*.
17. Choose *Continue.*

    The system displays the tree structure.
18. Choose *Edit* → *Expand*.

    The system displays the accompanying field objects for the table object.
19. Save your entries.

Result

You have assigned selected objects in your interface format to change validation, or created objects for the change validation.

Delimit Objects

Use

You can use the *Delimit* function for table objects [Page 27] (internal tables, field strings, and infotypes).

This function flags a table object and the accompanying field objects in the payroll period to be exported as invalid.

The *Delimit* function **cannot** be used separately for an individual field object. If you create this function for a table object, the system includes **all** field objects belonging to this object in the delimitation.

Features

By delimiting an object, **all field objects** belonging to this object are flagged with the character "<". The delimitation information **"<"** is transferred with the respective field object length.

The key fields [Page 75] are exceptions because the system always exports the contents of key fields.

During the data export by the Interface Toolbox, the delimited object is transferred to the third-party system along with all flagged field objects. By looking at the flagged field object the receiver can determine whether or not the object is still valid for the relevant employee.

Activities

Activate the Delimit [Page 71] function for a table object.

See also:

Example: Delimiting an Infotype [Page 70]

Example: Delimiting an Infotype

You have activated *Change Validation* for the *External Bank Transfers* infotype (0011). Choose *Delimit* and define a unique key for the infotypes, for example, the fields BANKL and BANKN.

The normal change validation process is not affected. However, if the infotype record with the defined key is delimited, the Interface Toolbox only defines the values for these key fields. The delimitation character "<" is entered in all other fields.

Your employee was a member of a club, and between January 1996 and March 1996 the membership fees were paid by external bank transfer. This amount is entered in *Ext.Bank Transfer (0011)* infotype.

The employee ended the club membership on March 31, 1996. You must now set the validity end date for this external bank transfer to March 31, 1996 using master data maintenance.

Use the Interface Toolbox to carry out the data export for April 1996. The following field objects for the *Ext. Bank Transfer (0011)* infotype are included in the objects transported to the third-party system.

External Transfers Infotype (0011)

Field Object	Name	Field content
P0011-BANKL	Bank key	672922
P0011-BANKN	Bank account number	700100
P011-BETRAG	Amount	<... <
P0011-LGART	Wage types	<... <

The field content "**<**" (delimitation) means that the *Ext.Bank Transfer* infotype (0011) and all relevant field objects are **invalid** for this employee in the export payroll period.

Delimiting Objects

Prerequisites

You have created an interface format.

Procedure

20. Choose the *Configuration* tab index.
21. In the *Object to be processed* group box, choose *Interface Format*.
22. In the *Interface format* field, enter the name of your interface format.
23. Choose *Change Validation.*
24. Choose *Display / Change*.

    The system switches to the change mode and displays the **tree structure**.
25. Choose *Edit* → *Expand*.

    The system displays the accompanying **field objects** for the **table object**.
26. Place the cursor on the table object.
27. Choose *Delimit.*

    The system delimits the selected table objects with the accompanying field objects.
28. Save your entries.

Result

You have delimited the table object and respective field objects for change validation.

You can reverse the delimitation by selecting the table object and choosing the *Delimit* function again.

Single Field Validation

Use

The *Single Field Validation* function only applies to table objects [Page 27].

The *Change Validation* function usually considers the field objects in a table object (internal tables, field strings and infotypes) as a unit. The system exports the complete table object as soon as a field object in the table object has changed.

The *Single Field Validation* function allows you to export just the field objects that have changed. If the field object has not changed, the “#” character is exported.

If no field object in a table object has changed from one payroll period to the next, then the **complete table object is ignored**. The corresponding table object is not exported.

Activities

Activate single field validation [Page 74] for a table object.

See also:

Example: Single Field Validation [Page 73]

Example: Single Field Validation

In the **TEST** interface format, the *Work Center/Basic Pay (WPBP*) table object is defined with the *Personnel Subarea (BTRTL), Personnel Area (WERKS)* and *Employee Subgroup Grouping for Personnel Calculation Rule (ABART)* field objects. The individual field validation function is activated for the table object (WPBP). The system exports the following values for the displayed payroll results:

Export with Single Field Validation

Object	Previous payroll result	Current payroll result	Data after import
WPBP-BTRTL	0001	0001	####
WPBP-WERKS	0002	0003	0003
WPBP-ABART	1	1	#

For Comparison:

Export Without Single Field Validation

Object	Previous payroll result	Current payroll result	Data after import
WPBP-BTRTL	0001	0001	0001
WPBP-WERKS	0002	0003	0003
WPBP-ABART	1	1	1

Validating Single Fields

Prerequisites

1. You have created an interface format [Page 30].
2. You have created the *change validation* for a table object [Page 68].
3. The *Maintain Change Validation* screen appears.

   The system displays the accompanying field objects for the table object [Page 71].

Procedure

1. Place the cursor on the table object.
2. Choose *Single Field Validation.*

   The system validates the single fields for the selected table object.
3. Save your entries.

Result

You have activated single field validation for a table object. In the future, only changed field objects will be exported.

You can reverse the single field validation by selecting the table object and choosing the *Single Field Validation* function again.

Key Fields

Use

You can only use the *Key Fields* function for field objects [Page 29] in internal tables or infotypes.

The *Key Field* function cannot be used for the **Field string** table object, since the field string only consists of one entry (for example, the *Personal Features (PERM)* field string in cluster object Rx (for example, RD, RU)).

Change validation for table objects (internal tables) can only function successfully if you specify which line of the current table is to be compared with which line in the table from the comparison period. The *Key Field* function defines those fields which clearly determine a table line as key fields for the table object. This means that one table cannot contain two lines that match in all key fields.

For each current table entry, the SAP System searches in the current payroll period for a table entry with the same key fields in the comparison period and then compares all fields.

Activities

Define the field objects as key fields [Page 77]

See also:

Example: Key Fields [Page 76]

Example: Key Fields

You have defined two field objects, *Bank Number (BANKL)* and *Account Number (BANKN)* as key fields in the table object containing the bank transfer amounts.

A bank transfer must exist for each account number so that change validation function can work with the key field function successfully.

Payroll Period 01	**Payroll Period 02**					
BANKL	BANKN	AMOUNT		BANKL	BANKN	AMOUNT
Key field	**Key field**			**Key field**	**Key field**	
672922	700100	900.00		672922	700100	950.00
672922	700101	500.00				
672922	700102	400.00		672922	700102	400.00
				672922	700103	100.00
				672922	600500	500.00

BANKL	**BANKN**	**AMOUNT**	**Legend**
Key field	Key field		
672922	700100	950.00	Change in amount
672922	700101	<…<	if delimitation active
672922	700103	100.00	Only in period 02
672922	600500	500,00	Only in period 02

The *Bank Number (BANKL)* and *Account Number (BANKN)* field objects are key fields. The content of these fields is exported, not the "**<**" character (delimitation).

Defining Key Fields

Prerequisites

1. You have created an interface format [Page 30].
2. You have created the *change validation* for a table object [Page 68].
3. The *Maintain Change Validation* screen appears.

   The system displays the accompanying field objects for the table object [Page 71].

Procedure

4. Place the cursor on the table object.
5. Choose *Key Fields.*

   The selected field objects in the tree structure have the key field symbol.
6. Save your entries.

Result

You have defined a key field object for change validation.

You can reverse the definition by selecting the field object and choosing the *Key fields* again.

Relations

Use

The *Relations* function only applies to field objects [Page 29].

To use the *Relations Between Field Objects* function, single field validation [Page 72] must be **active** for the relevant table object.

Features

The *single field validation* and *relations* are related as follows:

If single field validation is active for a table object, the system only exports fields from the table object that have changed.

There are often close relationships between individual fields in a table in the SAP System. This means that the fields are only meaningful **together**. For example, an employee's personnel subarea data is only meaningful if the personnel area to which the personnel subarea is assigned is also known.

To meet this requirement, you can use the Interface Toolbox to define **relations** between two or more fields. If at least one field that is linked to one or more other fields by the *Relations* function changes, then the system exports all linked fields again.

Activities

Create relations [Page 80] between field objects.

See also:

Example: Relations Between Field Objects [Page 79]

Example: Relations Between Field Objects

You have already activated *single field validation* for the *Work Center/Basic Pay (WPBP)* table object

You want to link the *Personnel Area (WERKS)* and *Personnel Subarea (BTRTL)* field objects using *relations*.

If the content of one of the two field objects changes from one data export to the next, the system exports both field objects again.

Previous payroll period	Current payroll period	Export the following data	Explanation
WPBP-BTRTL=0001	WPBP-BTRTL=0001	WPBP-BTRTL=0001	Export again
WPBP-WERKS=0002	WPBP-WERKS=0003	WPBP-WERKS=0003	Export again
WPBP-ABART=1	WPBP-ABART=1	WPBP-ABART=#	No change, single field validation is active

The system also checks field objects with no relations individually.

Creating Relations

Prerequisites

1. You have created an interface format [Page 30].
2. You have created the *change validation* for a table object [Page 68].
3. To perform this function, you must first activate Single Field Validation [Page 72] for the table object to which the field object belongs.
4. The *Maintain Change Validation* screen appears.

   The system displays the accompanying field objects for the table object [Page 71].

Procedure

1. Place the cursor on the selected field object.
2. Choose *Relations*.

   You access the *Relations for <name>* dialog box.
3. Select the field object(s) to be linked to the previously selected field object with a relation.
4. Save your entries.

Result

You have created relations for a field object in change validation.

If you choose the *Relations* icon in front of the name of the field object, you display a list of all field objects related to the selected field object.

To delete unwanted entries, select the entry and choose *Delete*.

Wage types

For information on the *Wage Types* function, see Wage Types in Change Validation [Page 96].

User-Defined Change Validation

Use

You can use this function to specify that the decision of whether or not to carry out change validation will only be made when the export program **runs**. If you want to use this function, the Toolbox requires a user-defined program to decide whether or not to activate *change validation*.

Features

The Interface Toolbox makes demands on the user-defined program so that the change validation can be activated when the program runs.

Program type

The **program type** can have the following values:

- **1** Executable program
- **S** Subroutine pool

Parameters

The program must contain a form routine with only **one parameter**. This parameter displays the return value for the form routine. The return value determines whether or not the export value is to carry out the change validation. The following values are possible for the return value:

- *Change validation* not active
- *Change validation* active

The *Change Validation* function is set to **active** as default.

INCLUDE

If you include INCLUDE 3 from the Export Program function (interface format [Page 21]) in the customer program, you can use current export data and the export data from the comparison period to the customer form routine to decide whether to activate change validation.

Activities

Activate the user-defined change validation [Page 85]

See also:

Naming Conventions for Export Data [Page 83]

Example: User-Defined Change Validation [Page 84]

Naming Conventions for Export Data

The following naming conventions apply to the export data:

- NEW_<table name> for the current export data (for example, NEW_WPBP)
- OLD_<table name> for the export data from the comparison period (for example, OLD_WPBP)

Example: Customer Program for Change Validation

The following example shows a possible structure for your user-defined program, which will be used to decide whether or not to activate change validation when the export program runs.

```
REPORT ZPCIFP01.
INCLUDE ZPCIFRX3.       "INCLUDE 3 from export program
FORM DIFFERENCE_CHECK USING SWITCH.
   SWITCH = `1`.         "change validation is active
   READ TABLE NEW_WPBP INDEX 1.
      IF SY-SUBRC = 0.
      READ TABLE OLD_WPBP INDEX 1.
      IF SY-SUBRC = 0.
         IF ( NEW_WPBP-MASSN <> OLD_WPBP-MASSN) AND
            ( NEW_WPBP-MASSN = `05`).
            SWITCH = `0`. "change validation is not active
ENDIF.
ENDIF.
   ENDIF
ENDFORM.
```

Legend

The form routine uses the SWITCH parameter, which decides whether or not the change validation is active.

The change validation function is set to active as default (SWITCH = 1). If the MASSN field in the *Work Center/Basic Pay* (WPBP) table for the current export data differs from that in the comparison period, and if the current personnel action (MASSN) is ‘05’, then change validation is not active.

Activating User-Defined Change Validation

Prerequisites

1. You have created an interface format [Page 30].
2. You have created the *change validation* for a table object [Page 68].
3. The *Maintain Change Validation* screen appears.

   The system displays the accompanying field objects for the table object [Page 71].

Procedure

5. Choose *User-defined change validation.*

   The *User-Defined Change Validation* dialog box appears.
6. In the *Program* field, enter the customer (user-defined) program. In the *Form Routine* field, enter the name of the accompanying form routine.
7. Save your entries.

Result

You have specified which user-defined program the Toolbox will use to decide whether or not to activate change validation when the export program is run.

Wage Type Processing with the Toolbox

Use

In payroll, wage types [Ext.] are important stores of information which are used to calculate wages and salaries. The wage type is particularly important in the Interface Toolbox, which was developed to export payroll results from the *Human Resources* (HR) component to a third-party system.

Special, enhanced processing rules apply for particular functions in the Interface Toolbox to meet the demands made on the wage types. These processing rules differ from the processing rules for individual functions. This concerns the following functions:

- Interface Format

  Wage Types in the Interface Format [Page 87]

- Change Validation

  Wage Types in Change Validation [Page 96]

  The same processing rules apply to wage types as to other objects for all functions in the interface format or change validation that are not specifically mentioned here (for example, the conversion function for field objects, and so on)

Wage Type Tables in the Interface Format

Definition

A *Human Resources* (HR) table can be treated as a wage type table in the interface format if the table contains at least **one field** with the **LGART** field name.

Use

The Interface Toolbox processes wage types and the accompanying wage type tables in the SAP System separately in the *interface format*.

Table Types

If there is a field in the table with the name LGART, then one of the following two **table types** can be set for this table:

- **Regular tables**

  The system treats the tables in the *interface format* as **regular tables** (for example, WPBP (*Work Center/Basic Pay*)).

- **Wage type tables**

  The system treats the tables in the *interface format* as **wage type tables** (for example, RT (*Results Table*)).

Assign Table Type

The *Restrictions* function in the interface format allows you to assign one of the two table types to a table containing at least one field with the name LGART. If you select one of the *special restrictions for wage types* using the Restrictions [Page 50] function, then the Toolbox will define this table as a **wage type table**.

The *Results Table (RT)* table object is automatically defined as a wage type table by the Interface Toolbox.

Other table objects are considered to be regular tables.

See also:

Wage Type Selection in the Interface Format [Page 88]

Wage Type Selection in the Interface Format

Use

You can select subsets of a wage type table (individual wage types) for export in the same way as when selecting a field object in a regular table. In both cases, use the Restrictions [Page 50] function in the *Interface Format* to select a subset of table entries (lines in the table [Page 89]) for wage types or field objects.

Different From Selecting Other Field Objects

The selection of wage types from wage type tables differs from the selection of other field objects form regular tables.

- If you **do not** enter any restrictions, all wage types will be exported.
- If you enter restrictions, the only wage types exported are those that you have assigned to the wage type table using the *Restrictions* function.

To process the wage types, the Interface Toolbox requires **additional information** that must be entered when the wage types are selected for data export (*Restrictions* function). This information determines how the wage types are processed in the original payroll run and also in retroactive accounting. This additional information is contained in the different **wage type options for retroactive accounting.**

If you have entered restrictions, the Interface Toolbox exports all wage types using wage type option [Page 90] R1.

If the same processing rules apply to a subset of wage types, then you can also select these wage types **generically**, for example, ***** or **/1***.

Prerequisites

To process wage type tables, you must assign your table objects to the *interface format* before you can export data.

See also:

Wage Type Options for Retroactive Accounting [Page 90]

Structure of a Wage Type

Definition

A wage type is a table entry in a wage type table. A wage type consists of the LGART field and the following three numerical fields:

- *Amount*
- *Rate*
- *Number*

Other fields also belong to a wage type (for example, *Split Indicator*, *Currency*).

Wage types	Amount	Rate	Number
LGART	(RTE)	(AMT)	(NUM)
For example, MA10	For example, 10	For example, 5	For example, 2

These fields form **one line** (table entry) in the wage type table for the corresponding wage type.

Wage Type Options for Retroactive Accounting

Use

The **ability to run retroactive accounting** is one of the most important characteristics of payroll. A retroactive run is necessary, for example, in the following case:

You have made changes to HR master data for the payroll past.

The system must run payroll for these payroll periods again, even though payroll is completed. The modified payroll data must be taken into account.

When comparing the payroll periods [Ext.] to be included in retroactive accounting with the original payroll period, the payroll program or reporting programs may identify that corrections are required.

As a result of a retroactive salary increase, an additional amount must be transferred to the employee.

Wage Types in a Retroactive Run

Most wage types are created in the retroactive accounting period with the amount that would have been used in the original period (for example, wage type */101 (Total Gross Amount)).*

However, there are wage types that contain the differences between the retroactive period and the original payroll period.

Wage type */553 (Recalculation difference for last payroll run)* is the difference between the payment amount for the current payroll period in the payroll run and the previous result.

To support third-party payroll systems that do not use retroactive accounting, or that use a different method to form retroactive accounting differences, the Interface Toolbox has **retroactive accounting functions.**

Features

- **Retroactive Accounting Option R1**

  The system processes the selected wage type **as in the original payroll period**. The system transfers the values in the three wage type fields (*Amount*, *Rate*, and *Number*) **unchanged** to the third-party system.

- **Retroactive Accounting Option R2**

  The system calculates the **difference** between the values in the three wage type fields (amount, rate, and number) for the selected wage type from the retroactive accounting period and the comparison period [Page 92]. These differences are **assigned** to the respective **retroactive accounting period**. If the difference for all three fields is **0** (zero) then the system does not export this wage type.

- **Retroactive Accounting Option R3**

As for option R2, the system calculates the **difference** between the values in the three wage type fields (amount, rate, and number) from the retroactive accounting period and the comparison period. However, these differences are assigned to the period immediately preceding the current payroll period.

- **Retroactive Accounting Option R4**

  As for options R2 and R3, the system calculates the **difference** between the values in the three wage type fields (amount, rate, and number) from the retroactive accounting period and the comparison period. These differences are assigned to the **current payroll** period. If the relevant wage type already exists in the payroll period, then the system cumulates the values for each personnel number. If the third-party system does not use retroactive accounting, you should choose this wage type option.

- **Option DT**

  Options R3 and R4 have a special feature to allow wage types to be assigned to a payroll period, even if they do not originate in this payroll period and come from retroactive accounting periods instead. If you wish to retain information on whether the wage type comes from a retroactive accounting period or from an original period, then choose option DT.

  The system assigns wage types that originate from retroactive accounting to a table with the name $D<name>, instead of to a table in the original period.

Activities

Choose a wage type option [Page 95]

Comparison Period for Wage Type Options in Retroactive Accounting

Definition

The comparison period is a payroll period used within the *Restrictions* function in *Change validation*. It is used to determine which data has changed since the last export.

Use

If you are running first retroactive accounting for the first time for the for-period [Ext.] in question, the Interface Toolbox will use the original payroll period as the comparison period. However, it could be the case that **several** retroactive accounting runs are required for a payroll period [Ext.]
.

You are recalculating payroll period **01** from period **02**.

If it is necessary to recalculate payroll period **01** from period **04** again, then the differences are created as follows.

The difference between the payroll period **"period 01 recalculated from period 04"** and the payroll period **"period 01 recalculated from period 02".**

The difference for the original period 01 is **not** created.

Rule

The payroll period that the system always uses for comparison is the most recent payroll period with the same for-period view [Page 93] as the payroll period currently being processed.

See also:

Defining the Comparison Period for Generating Wage Type Differences for Several Retroactive Runs [Page 94]

In-Period Information / For-Period Information

The **for-period information** specifies the payroll period **for** which the payroll result is valid.

The **in-period information** specifies the payroll period **in** which the system has created the payroll result.

Together the **in-period information** and the **for-period information** characterize **one** payroll result.

Example: For-Period Information / In-Period Information

Payroll result	For-per. information	Start date of for-per. information	End date of for-per. information	In-per. information	End date of in-per. information
1	01/1997	01.01.1997.	31.01.1997	02/1997	02.28.1997
2	02/1997	02.01.1997	28.02.1997	02/1997	02.28.1997

Explanation

Payroll result 1 for payroll period (for-period) 01/1997 has the start date January 01, 1997 and the end date January 31, 1997. This payroll result was created in payroll period (in-period) 02/1997.

Example: Comparison Period for Wage Type Differences for Several Retroactive Runs

Do not confuse this example with the determination of the comparison period for change validation [Page 58].

An employee has the payroll results listed in the table. The payroll results contain each relevant payroll comparison period for the creation of wage type differences (changes). The system compares the most recent payroll period with the same for-period information [Page 93] as the payroll period being processed.

Sequential no.	For-period	In-period	Preceding period
00001	01 1996	01 1996	No preceding period
00002	01 1996	02 1996	00001
00003	02 1996	02 1996	No preceding period
00004	03 1996	03 1996	No preceding period
00005	01 1996	04 1996	00002
00006	02 1996	04 1996	00003
00007	03 1996	04 1996	00004
00008	04 1996	04 1996	No preceding period

Activating Wage Type Options for Retroactive Accounting

Prerequisites

1. You have created an interface format [Page 30].
2. You are in the object tree [Page 32] and all table objects are displayed with all field objects.

Procedure

10. Place the cursor on the selected wage type field (LGART) in the table object to be edited.
11. Choose *Restrictions.*

    You access the *Set Restrictions* dialog box.
12. Select *Special restrictions for wage types*.

    You access the *Wage Type Restriction for <name>* dialog box.
13. Enter the restrictions.

    Several entries for a field object are linked implicitly with the logical operation **OR.**
14. Enter the name of the wage type and select the required option (DT, R1-R4).
15. Choose *Transfer.*
16. Save your entries.

Result

You have used the Interface Toolbox to activate one of the wage type options for retroactive accounting (R1 to R4) for a wage type field. If you have already activated wage type option DT, wage types used in the retroactive run will be assigned to the table $D<table name >. The information necessary for the data export of a wage type is now complete and the wage type is assigned to the corresponding wage type table.

Wage Types in Change Validation

Use

The Interface Toolbox processes wage types and the accompanying wage type tables separately during change validation.

Just as the Interface Toolbox processes regular tables [Page 87] in change validation, you can also run change validation [Page 56] for wage type tables. The system only exports the wage types to the third-party system if they have changed from one data export to another.

The following topics are important for using this transaction:

- Change validation and wage type tables [Page 97]
- Change validation and wage type comparison [Page 98]
- Wage types and split indicators [Page 99]

Features

The following functions are available for wage types in change validation:

- Activate wage types for change validation [Page 100]

  The functions are only valid for wage types.

You can use the following change validation functions to process wage types. The functions have modified or restricted processing rules.

- Delimit wage types for change validation [Page 102]

The following functions **cannot** be used in connection with the change validation:

- *Validating single fields*
- *Key fields*

  The explicit definition of key fields is not possible for wage types. The wage type name **(LGART)** is always used as the key field.
- *Relations*

Prerequisite

You have assigned the individual wage type to the relevant interface format using the Restrictions [Page 50] function for field objects in the *Interface Format*. In this assignment, you have activated one of the wage type options for retroactive accounting (R1 to R4) for each wage type to be included in change validation.

If you have not entered restrictions, all wage types will be exported with wage type option R1.

Change Validation and Wage Type Tables

If it is a regular table, activate change validation for the complete table.

If it is a wage type table, then change validation always refers to a **single wage type**.

Just as with regular tables, the only field objects included in change validation are those you have already assigned to change validation. You must assign each wage type required for the difference check to change validation.

If you do not specify any wage types. change validation will **not** take place in the standard system.

Change Validation and Wage Type Comparison

You can set the comparison period for change validation with wage types as for all other objects. When the system checks a wage type [Page 89] for changes, it uses the three fields (amount, rate, and number) that are common to all wage types.

The wage type comparison has the following results:

- If the content of at least one of the three wage type fields has changed then the relevant wage type and the three fields will be **exported** again.
- If the content of none of the three wage type fields has changed then this wage type will be ignored. The relevant wage type is **not exported**.

For more information on comparison periods in change validation, see Comparison Periods in Change Validation [Page 58].

Wage Types and Split Indicators

If wage types have split indicators [Ext.], there can be several table entries (lines in the table [Page 89]) for this wage type in the wage type table. As a result the wage type name is **not** sufficient as a key to identifying a table entry.

The Problem of Wage Types and Split Indicators [Ext.]

In change validation, it is not clear which table entry for the wage type in the comparison period is to be compared with which table entry in the current payroll period.

Solution

If you want to use change validation for wage types, you must ensure that only **one table entry exists per wage type**. To ensure that this is the case, **do not export split indicators**. You must not select the field object containing the split indicator for export. If you have already selected this field object for export, **delete** it. If the wage type exists several times after the split indicator has been deleted, the system cumulates the multiple entries for this wage type and stores them in one entry.

Activating Wage Types for Change Validation

Use

The *Wage Types* function is only allowed for wage types. The system only checks the wage types for changes if you have assigned the wage types to change validation using the *Wage Types* function.

Activities

Activate wage types for change validation [Page 101]

Activating Wage Types for Change Validation

Prerequisites

1. You have created an interface format [Page 30].
2. You **must** first use the Restrictions [Page 50] function to assign the wage type to be processed to the interface format to be processed.
3. You have created the *change validation* for a table object [Page 68].
4. The *Maintain Change Validation* screen appears.

   The system displays the accompanying field objects for the table object [Page 71].

Procedure

29. Place the cursor on the field object **LGART**.
30. Choose *Wage types*.

    You access the *Wage Types for <name>* dialog box.
31. Select the wage types you want to include in change validation.
32. Choose *Continue.*
33. Save your entries.

Result

You have assigned individual wage types in a wage type table to change validation.

You can remove individual wage types from the assignment by selecting the wage type symbol. The system displays the wage types. Select the wage type and choose *Delete.*

Wage Type Delimitation for Change Validation

Use

The *Delimit* function refers to a single **wage type**. A wage type table is delimited on the wage type level and must be carried out for each wage type **individually**.

Alternatively, you can use the *Delimit* function following the instructions under Delimiting Objects [Page 69].

Constraints

If you use the *Delimit* function for wage types, you should note that a wage type is delimited irrespective of the wage type options for retroactive accounting (R1 to R4).

Activities

Delimit wage types for change validation [Page 103]

Delimiting Wage Types for Change Validation

Prerequisites

1. You have created an interface format [Page 30].
2. You **must** first use the Restrictions [Page 50] function to assign the wage type to be processed to the interface format to be processed.
3. You have created the *change validation* for a table object [Page 68].
4. The *Maintain Change Validation* screen appears.

   The system displays the accompanying field objects for the table object [Page 71].

Procedure

1. Choose wage types.

   The system displays the accompanying field objects for the **LGART** field.
2. Place the cursor on a wage type.
3. Choose *Delimit.*

   The Delimit symbol appears next to the wage type.
4. Save your entries.

Result

You have delimited individual wage types in a wage type table for the data transport.

Examples: Interaction of Wage Types and Wage Type Options for Retroactive Accounting in Change Validation

The following example explains how the individual change validation functions can be used for wage types from wage type tables.

Example: An employee has the following payroll periods

Sequential no.	For-period	In-period	*Total Gross Amount* wage type from the results table (RT)
00001	01 1996	01 1996	3 000.00
00002	02 1996	02 1996	3 000.00
00003	01 1996	03 1996	3 500.00
00004	02 1996	03 1996	3 500.00
00005	03 1996	03 1996	3 500.00
00006	03 1996	04 1996	Wage type not available
00007	04 1996	04 1996	Wage type not available

The following examples consider the generation of a special wage type. The *Total Gross Amount* wage type (secondary wage type /101) from the *Results Table (RT)* is used as an example wage type for the employee.

The *total gross amount* wage type contains an amount of **3,000.00** for the payroll periods **01** and **02.**

- Before payroll is run for payroll period **03**, the *Total Gross Amount* wage type is increased retroactively to **3,500.00** as of payroll period **01**. This increase corresponds to a salary increase for the employee. The retroactive increase in the total gross amount triggers retroactive accounting for this employee.
- **Before** the payroll run for payroll period **04,** the *Total Gross Amount* wage type is set to **ZERO** (0) as of payroll period **03** since the employee should not receive any pay as of payroll period **03.**

  Payroll is run for payroll period **04** with retroactive accounting for period **03**.

The following examples show how the Interface Toolbox works in connection with wage type processing in the interface format and change validation.

Examples: Interaction of Wage Types and Wage Type Options for Retroactive Accounting in Change V

Example 1: Third-Party Payroll System Runs Retroactive Accounting (R1)

A third-party payroll system runs retroactive accounting. You want to transfer the *Total Gross Amount* wage type and all other wage types in the SAP System from the employee's payroll result to the third-party system.

To do this, choose retroactive accounting option R1 [Page 90]. The *Total Gross Amount* wage type has the following values after the data export:

Payroll period	***Total Gross Amount* wage type from the results table (RT)**
01 in 01	3 000.00
02 in 02	3 000.00
01 in 03	3 500.00
02 in 03	3 500.00
03 in 03	3 500.00
03 in 04	----------
04 in 04	----------

Legend

1. In the **01 in 03** and **02 in 03** payroll periods, the system exports the unchanged total values for the wage type from the original period.
2. The wage type is no longer available in the payroll periods **03 in 04** and **04 in 04** since it was set to **ZERO** as of payroll period **03**, before the payroll run for payroll period **04**.

Example 2: Third-Party Payroll System Runs Retroactive Accounting (R1); Change Validation and Delimitation Functions are Active

You only want to export wage types if they have changed from one payroll period to another. You activate change validation for your wage types by assigning single wage types to change validation using the *Create* function. (For more information on creating objects in change validation, see Creating Objects [Page 67]).

The following **problem** can occur in this situation.

A wage type from a payroll period does **not** exist in the export file. It is unclear whether the wage type did not occur in this payroll period, or whether it has a value **identical** to the value in the comparison period.

There are two possible solutions to this problem:

- The user is active:

  You provide information on whether the wage type still exists in the payroll period. You can do this using an additional wage type. This additional wage type acts as a switch (switch wage type). The system generates the switch wage type if the original wage type no longer exists.

- The Interface Toolbox is active with the *Delimit* function:

  You can delimit the wage type using the corresponding function. The system generates the character “<“ for the wage type in question. The delimitation sign means that the wage type is no longer valid for the employee.

For more information on delimiting objects within change validation, see *Change Validation → Delimiting Objects* and *Wage Type Processing in the Interface Toolbox → Delimiting Wage Types for Change Validation.*

If change validation and delimitation are active for the wage type, then the *Total Gross Amount* wage type has the following result data for the data export:

Payroll period	***Total Gross Amount* wage type from the results table (RT)**
01 in 01	3 000.00
02 in 02	----------
01 in 03	3 500.00
02 in 03	----------
03 in 03	---------
03 in 04	< ... <

04 in 04	----------

Legend

1. The wage type in the **02 in 02** payroll period is not exported since the total value of the wage type has **not** changed when compared with the values in the **01 in 01** payroll period.
2. The wage type is not exported in the **02 in 03** and **03 in 03** payroll periods, since the total value for the wage type has **not** changed from the value in the **01 in 03** payroll period.
3. In the **03 in 04** payroll period, the wage type is delimited, since it no longer exists.
4. The wage type no longer exists in the **04 in 04** payroll period since the system has already transferred the delimitation information to the **03 in 04** payroll period

Example 3: Third-Party Payroll System Runs Retroactive Accounting (R2)

A third-party payroll system runs retroactive accounting. You only want the SAP System to transfer the differences between the payroll period used in retroactive accounting and the payroll period from the original payroll run to the third-party system for retroactive accounting periods.

To do this, choose retroactive accounting option R2 [Page 95].

The *Total Gross Amount* wage type has the following results data for the data export:

Payroll period	***Total Gross Amount* wage type from the results table (RT)**
01 in 01	3 000.00
02 in 02	3 000.00
01 in 03	500.00
02 in 03	500.00
03 in 03	3 500.00
03 in 04	-3 500.00
04 in 04	----------

Legend

1. In the **01 in 03** and **02 in 03** payroll periods, the system only exports differences from the retroactive accounting period and the original period.

   3 500.00 **-** 3 000.00 = 500.00

2. In the **03 in 04** payroll period, the total wage type amount is - 3,500.00, since the value of the wage type was retroactively set to ZERO in payroll period **03**.

   0 - 3 500.00 = **-** 3 500.00

3. The wage type no longer exists in the **04 in 04** period, since it was set to ZERO as of payroll period **03**, before the payroll run for payroll period **04**.

Example 4: Third-Party Payroll System Runs Retroactive Accounting (R2); Change Validation and Delimitation Functions are Active

A third-party payroll system runs retroactive accounting. You only want the R/3 System to transfer the differences between the retroactive accounting payroll period and the comparison period to the third-party system during retroactive accounting periods.

For more information on the comparison period, see *Defining the Comparison Period for Wage Type Options in Retroactive Accounting.*

To do this, choose retroactive accounting option R2 [Page 90].

By activating change validation, you also ensure, as in example 2, that export wage types are only exported again if they have **changed**.

Make sure that change validation is only activated for the **original periods,** and not for the retroactive periods in payroll.

The change validation function for retroactive accounting periods is **controlled** by the R2 retroactive accounting option.

Change validation for the original periods in payroll also refers to the total value of the respective wage type and not to the differences.

To delimit wage types, follow the procedure recommended in example 2.

The system only delimits wage types if the respective wage type exists and has a difference in retroactive accounting.

The *Total Gross Amount* wage type has the following results data for the data export:

Payroll period	***Total Gross Amount* wage type from the results table (RT)**
01 in 01	3 000.00
02 in 02	----------
01 in 03	500.00
02 in 03	500.00
03 in 03	----------

03 in 04	-3,500.00 (first entry) < ... **<** (additional delimitation information**)**
04 in 04	----------

Legend

1. The wage type in the **03 in 03** payroll period is **not exported** since the **total value of the wage type** has not changed when compared with the values in the **02 in 03** payroll period.
2. In the **03 in 04** payroll period, the results table (RT) contains **two** entries for the *Total Gross Amount* wage type. On the one hand, the difference between the original **03 in 03** payroll period and **03 in 04** (0.00 - 3,500.00 = -3,500.00) is transferred. On the other hand, the wage type no longer exists in the payroll period, so the **"<"** delimitation information must be transferred.
3. The wage type no longer exists in the **04 in 04** payroll period since the delimitation information has already been transferred in the **03 in 04** payroll period

Example 5: Third-Party Payroll System Without Retroactive Accounting (R4)

The third-party system performs payroll **without** retroactive accounting.

Therefore, you want to **collect** the differences from the retroactive accounting run and transfer them with the corresponding original period to payroll.

To do this, choose retroactive accounting option R4 [Page 90].

The *Total Gross Amount* wage type has the following results data for the data export:

Payroll period	***Total Gross Amount* wage type from the results table (RT)**
01 in 01	3 000.00
02 in 02	3 000.00
01 in 03	----------
02 in 03	----------
03 in 03	4 500.00
03 in 04	----------
04 in 04	3 500.00 (Repayment from 03 as recalculation difference)

Legend

1. The differences from the **01 in 03** (500.00) and **02 in 03** (500.00) payroll periods are collected.

   The value of 4,500.00 is calculated as follows:

   2 ***** 500.00 **+** 3 500.00 **=** 4 500.00

2. The wage type is no longer available in the payroll periods **03 in 04** and **04 in 04** since it was set to **ZERO** as of payroll period **03**, before the payroll run for payroll period **04**.

Example 6: Third-Party Payroll System Without Retroactive Accounting (R4); Change Validation and Delimitation Functions are Active

The third-party system performs payroll **without** retroactive accounting. Therefore, you want to **collect** the differences from the retroactive accounting run and transfer them with the corresponding original period to payroll. To do this, choose retroactive accounting option R4 [Page 90].

You also want to activate the change validation and delimitation functions for wage types. This can cause problems, since the information (wage type) is condensed in such a way that the result could be interpreted incorrectly.

The *Total Gross Amount* wage type has the following results data for the data export:

Payroll period	*Total Gross Amount* wage type from the results table (RT)
01 in 01	3 000.00
02 in 02	---------- (since change validation is not active)
01 in 03	----------
02 in 03	----------
03 in 03	4 500.00
03 in 04	----------
04 in 04	-3,500.00 (first entry) < ... **<** (additional delimitation information**)**

Problem

- If this information is the **only** information available for the wage type, then it could be interpreted as follows:
- In payroll periods **01** and **02** the wage type was worth 3,000.00.
- This value did **not** change with retroactive effect in the **01 in 03** and **02 in 03** payroll periods.
- In payroll period **03**, the wage type was worth 4,500.00.

  The calculation of the amount 4,500.00 (2*500.00 **+** 3,500.00 **=** 4,500.00) cannot be reconstructed here. However, it is necessary since change validation is used.
- This would also mean that the value of the wage type had decreased retroactively for payroll period **03**, and would have the value 1,000.00.
- Payroll period **04** is the first period in which the wage type does not occur.

Examples: Interaction of Wage Types and Wage Type Options for Retroactive Accounting in Change Validation

Solution

To solve this problem, you should activate the DT [Page 90] wage type option for retroactive accounting. The values for the *Total Gross Amount* may appear as follows:

Payroll period	*Total Gross Amount* wage type from the results table (RT)	*Total Gross Amount* wage type from the table ($DRT)
01 in 01	3 000.00	----------
02 in 02	----------	----------
01 in 03	----------	----------
02 in 03	----------	----------
03 in 03	3 500.00	1 000.00
03 in 04	----------	----------
04 in 04	< ... <	-3 500.00

Legend

1. In the **03 in 03** payroll period, the retroactive accounting difference of 1,000.00 and the amount of 3,500.00 are shown separately. This clearly shows that the current value of the wage type is 3,500.00 for the **03 in 03** payroll period.

   This information would be of particular importance for the payroll run in the **04 in 04** payroll period if the wage type had retained the value of 3,500.00 and no longer occurred because the change validation function was active. It would be unclear whether the employee is entitled to 3,000.00 or 4,500.00.

2. The change validation function sends the delimitation information to the “04 in 04” payroll period even though the wage type no longer exists in the **03 in 04** payroll period.

   The reason for this is the *Only Export in Original Period* option which is flagged in the selection screen for the export program.

   If you had not selected this option, then the Toolbox would have sent the delimitation information in the **03 in 04** payroll period.

Generation - Interface Format for the Export Program

Use

The Interface Toolbox does not use a standard data export program when exporting data. This means that there is no ready-made program for data export in the SAP System.

Features

The following demands are placed when exporting data from the SAP System to a third-party system.

- Flexibility for the user
- High transfer speed during data export

To meet these requirements, the system generates the export program based on user's requirements. These user requirements are stored in the tables in the Interface Toolbox.

If you have created an interface format, or have changed one of the following areas, you **must** regenerate the export program when before you can use it:

- Interface format [Page 21]
- Import macro for cluster data [Ext.]
- Data definition include [Page 53]
- Change validation [Page 56]
- Wage type processing [Page 86]

Prerequisite

Before you generate the export program, you must enter the name of the export program in the interface format for the Toolbox:

- In the *Main Program* field, enter the name of the export program
- In the *Include 1 to 9* fields, enter the name of the data definition include

Do not use the name of a program or include that has already been used for a different purpose.

The Toolbox only receives the names of the export program and includes. The coding for the export program is generated in the interface format via the *Generate* function.

Activities

Generate the export program [Page 116]

Generating the Export Program

Prerequisites

You have created an interface format or made changes in one of the following areas:

- Interface format [Page 21]
- Import macro for cluster data [Ext.]
- Data definition include [Page 53]
- Change validation [Page 56]
- Wage type processing [Page 86]

Procedure

13. Choose the *Configuration* tab index.
14. In the *Object to be processed* group box, choose *Interface Format*.
15. In the *Interface format* field, enter the name of your new interface format.
16. Choose *Generate.*

Result

The customer export program for the interface format has been generated for data export.

Export Program

Definition

Using your defined interface format as a basis, the Interface Toolbox generates a program using the *Advanced Business Application Programming* language (ABAP). This program is the export program.

Use

The export program transfers payroll results from the SAP System to a third-party system. You must

- generate [Page 115], and
- execute [Page 118] the export to suit your requirements.

Structure

When you start the export program, you see a selection screen with several input blocks.

- You can select the payroll period and personnel numbers in the upper areas.
- The lower area allows you set parameters for the export program.

Activities

Start the export program [Page 118]

Starting the Export Program

Prerequisites

You have created an interface format.

Procedure

1. Choose the *Export* tab index.
2. In the *Export activities* group box, choose *Export with interface format*.
3. In the *Interface format* field, enter the name of your interface format.
4. Choose *Execute*.

   You access the *Export* dialog box.
5. In the individual fields, enter the necessary data and select the required export options.
6. Choose *Execute*.

Result

You have created the export file for the personnel numbers selected from a payroll area.

Infotype: Export Status (0415)

Use

The Interface Toolbox uses the *Export Status* infotype (0415) to monitor the export status for the individual interface formats.

The infotype enables you to run payroll at the same time as other interface formats (Master Data interfaces and payroll interfaces). Separate administration of retroactive accounting (using the *Payroll Status* infotype (0003)) and interfaces (using the *Export Status* infotype (0415) subtype (interface name)) ensures that unwanted interaction is avoided. The retroactive accounting capabilities remain unrestricted.

Structure

Subtype

- Interface name

  Name of the interface format from which the export program has been generated.

Fields

The fields correspond to the relevant fields in the *Payroll Status* infotype (0003).

- *Earliest personal RA date*

  Earliest date from which data can be exported.

- *Export until*

  The employee data is exported until this date, even if the date is after the employee leaving date. The date must be in a period during which the employee is not active.

- *Do not export after*

  The employee data was exported until this date.

- *Exported until*

  The employee data has been exported up to this date.

- *Earliest MD change*

  If payroll data is changed for an employee, the R/3 System stores the validity start date.

  The export program uses this date to identify whether an export must be repeated, and if so, the date on which the new export must take place to transfer the change to the third-party system.

- *Personnel number locked*

  If this field is flagged, the employee is locked for export and the personnel number will not be exported. In this case, the personnel number is not exported.

Features

The Interface Toolbox creates and maintains the *Export Status* infotype (0415).

Fields

- The interface format creates the *Export Status* infotype (0415) for this personnel number when the personnel number is exported for the first time with a particular interface format.
- For each subsequent export, the Interface Toolbox enters the end date of the exported payroll period in the *Exported Until* field.
- The SAP System automatically enters the appropriate date in the *Earliest MD Change* field at the same time as the *Earliest MD change* field in the *Payroll Status* infotype (0003).
- You can use the remaining fields to monitor the export of individual personnel numbers. The Interface Toolbox usually maintains the *Export Status* infotype (0415) for you.

Export History for Interface Results

Use

To carry out detailed change and wage type validation, the export program must be able to access the data from previous exports. For this reason, the Interface Toolbox stores the data for each export.

Just like the payroll results, the stored interface results can be divided as follows:

- Interface directory

  The interface directory is a table of contents for personnel numbers. It contains all payroll runs, along with other detailed information (for example, for-period, in-period, payroll area.)

- Interface data

  The interface data contains the individual export data for the personnel number(s).

  The interface data is always in its original state, that is, the state before change validation is carried out.

The Interface Toolbox can work with several interface formats. As a result, the interface results are stored for each interface format.

Prerequisites

You must select the *Update* export option before you start the export program [Page 117].

Features

You can use the Toolbox to

- Display
- Delete

  It is often a good idea to delete interface results in the test system. The Interface Toolbox offers you two options:

  - Delete the last interface result for one or more personnel number(s).

    The interface results for the selected personnel number(s) are deleted.

  - Delete all interface results for an interface format.

    If no personnel number is entered, the interface results for all personnel numbers belonging to this interface format are deleted.

  To ensure consistency, it is not possible to delete an individual interface result within the sequence of interface results. The last (current) interface result is the only exception.

Displaying the Interface Format

Prerequisites

You want to display the interface results for an interface format after the data export.

Procedure

1. Choose the *Export History* tabstrip.
2. In the *Interface format* field, enter the name of your interface format.
3. If required, you can use the *Personnel no.* and *Payroll area* fields to restrict the selection of the data.
4. Choose *Display*.
5. Select a line from the interface results.

   The system displays the content of the table at the time when the selected interface result was exported.

   To give a complete overview, both the exported entries and those entries not exported due to change validation are displayed. However, wage types with restrictions are not displayed.

Result

You have displayed the interface data for an interface format.

Deleting Interface Results

Prerequisites

You want to display the interface results for an interface format after the data export.

Procedure

1. Choose the *Export History* tabstrip.
2. In the *Interface Format* field, enter the name of your interface format and, if required, the personnel number(s).
3. Choose *Delete*.

   You access the *Delete Interface Results* screen.
4. If you want to select individual lines in the interface result for deletion, flag *Manual selection.*
5. If you want to delete all the interface results, flag *Delete all to.*

   Enter the date up to which you want to delete the interface results.
6. Choose *Delete*.

Result

You have deleted interface results for an interface format.

Automatic Conversion of Interface Results

Use

The Interface Toolbox saves all values for all exported fields as interface results in cluster IF. The Interface Toolbox uses this information to determine whether the field has changed since the last export. This takes place in change validation. Changes in the data structures (field length or field type in the Data Dictionary) may mean that the Interface Toolbox cannot import results when exporting or displaying the interface results.

If a result with an old data structure cannot be imported, the Interface Toolbox temporarily converts the data to be imported to the new format. This is done using the data definition in the *Version Management Cluster IF - Exported Tables* (T532K) and *Version Management Cluster IF - Fields Structure* (T532L) tables.

The Interface Toolbox runs this data conversion automatically when the old results are imported. No further processing by the user is required.

The data is stored in the database in the old data structure. This avoids the loss of information.

Upgrade to Release 4.5

There is no version description for the old version for an upgrade from a previous release. If data structures are changed in an upgrade, you must carry out one-time activities.

For more information, see the documentation for report RPCLSTIF_CONVERSION (*Conversion of Interface Results in IF Cluster*) under *Structure Changes for an Upgrade*. When you call the report, choose *Help* → *Application help*.

Manual Conversion of Interface Results

Use

In addition to automatic conversion of the interface results without permanent conversion on the database, the Interface Toolbox also enables you to convert and store interface results manually. You use manual conversion to

- Run a test
- Quickly process old interface results that must be imported frequently (performance)

Prerequisites

You have used the Interface Toolbox to create an interface format and used this format to export payroll results or HR master data. The interface results have been saved in cluster IF. One of the exported data structures was then changed in the Data Dictionary.

You, or SAP, have changed the type or length of a data element.

If you want to import data from the old interface format, you must transfer the data to a new format.

You want to run a conversion of the old interface results on the database, and do not want to convert the interface results automatically [Page 124] .

Features

Manual conversion takes place for a particular interface format, a selected old and new version of the data, and a selection of personnel numbers.

For more information, see the documentation for report RPCLSTIF_CONVERSION (*Conversion of Interface Results in IF Cluster*). When you call the report, choose *Help* → *Application help*.

Displaying TemSe Files

Prerequisites

You have exported the data.

Procedure

1. In the menu, choose *File* → *Display TemSe object.*

   The *Display TemSe object* dialog box appears.

2. In the *TemSe object* field, enter the names of your export field with the appropriate prefix.

   You want to display the export file with the name **`TESTFILE`**, which you have selected from the export program selection screen.

   In the *File* field, enter HR_PINTFS_TESTFILE and choose *Display.*

3. Choose *Display*.

   If the Toolbox identifies an error when export file is displayed, it will display the export file in an unformatted form [Page 161]. The unformatted display is better for identifying errors.

Result

The content of the TemSe file is displayed.

Managing TemSe Files

Use

You want to use the enhanced TemSe management functions:

- Delete TemSe objects
- Display TemSe files [Page 126]

Prerequisites

You have exported the data and created a TemSe object.

Procedure

1. In the menu, choose *File* → *TemSe administration.*

   You access the *TemSe: Request Screen*.

2. Enter the required selection criteria.

   If you do not know the exact object name or creator, leave the fields blank and enter an approximate date in the *Creation date* field. Try to enter a date that is as accurate as possible.

3. Choose *Object list.*

   You access the *TemSe: List of Objects* dialog box.

4. To display the contents of an object, select the required object and choose *Contents.*

   You access the *TemSe: Contents of Object* screen.

5. Different selection options for the selected objects are now displayed.

Result

You have accessed the TemSe file administration functions.

Downloading an Export File

Use

You can download the exported file from the TemSe file to your PC or application server.

Prerequisites

The data was successfully exported.

Procedure

1. Choose the *Export* tab index.
2. In the *Export activities* group box, choose *Download from TemSe*.
3. In the *TemSe object* and *File name* fields, make the appropriate entries.
4. Choose *Execute*.

Result

The exported TemSe file can be found on your PC or application server.

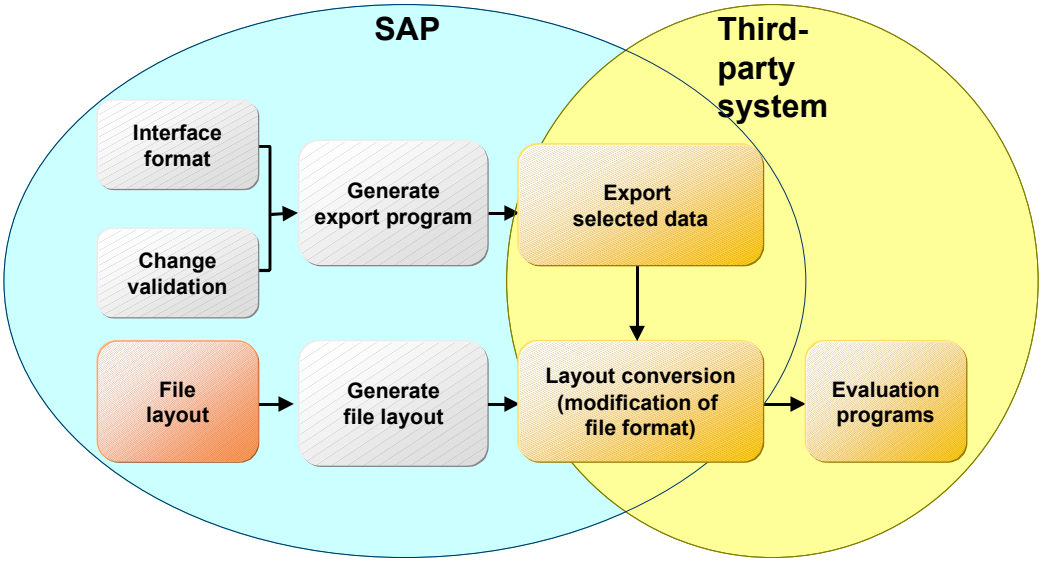
File Layout

Definition

The file layout describes the structure of the export file, in particular the sequence and length of the field objects [Page 32].

Use

The *File Layout* function allows you to convert the export file format, which is known as the SAP export file format, into any user-defined file format. An interface format [Page 21] is assigned to each file layout.



File Layout Types for the Export File

- SAP standard file layout [Page 162]
- User-defined file layout

The section on the file layout refers to the user-defined file layout.

Structure

Like the interface format, the file layout has hierarchical objects.

Objects in the File Layout and their Hierarchy

File Layout

Object	Hierarchy classification	Example
Block	1	BEGIN_OF_FILE
Structure	2	MASTER_DATA
Field	3	P0002-NACHN

You can define **blocks**, **structures** and **fields** in the file layout. When you create a file layout, use the following sequence:

1. Blocks
2. Assigned structures
3. Fields of the assigned structures

The blocks, structures and fields can have any name.

Fields

- You can assign fields in the interface format to the defined fields belonging to the file layout.

  You can also copy the characteristics and content of these interface format fields.

- You can also define the fields to suit your requirements.

Structures

You can group a certain number of fields together in a structure. A structure corresponds to one **line** in the export file.

Blocks

Several structures are grouped together in a block. Blocks are not represented in the export file. They are only used to control the editing of the structures.

See also:

Generating the File Layout [Page 156]

Layout Conversion [Page 158]

Processing the File Layout

Purpose

You use this process to convert the file format for the export file to a user-defined file format.

Prerequisites

The SAP System contains an export file in the SAP standard file layout [Page 161].

Process flow

1. Creation of file layout [Page 154].

   The defined blocks, structures, and fields form the file layout for the export file.

2. Generation of file layout [Page 156]

   You must also generate the file layout at the same time as the export program.

3. Conversion of file layout [Page 158].

   The Conversion function uses the rules you have defined to process the blocks, structures, and fields. The blocks, structures and fields are processed **individually** and processing is triggered by specific processing times in the Interface Toolbox. The Interface Toolbox calls these processing times, and processes all blocks that you have assigned to this processing time. The Interface Toolbox subsequently processes all structures assigned to the block, and all fields assigned to the structure.

   If the file layout function does not meet your requirements, you can define form routines to process the file layout and call these form routines using a user exit.

Result

The SAP System contains an export file with the user-defined file layout.

Editing and Attributes for the File Layout

Use

You use these functions to edit the file layout.

Features

- Create

  The *Create* function is valid for all objects (block, structure, field).

  You **must** assign an existing **interface format** to the file layout.

  - If the file layout does not yet have any objects (block, structure, field), choose *Create*.
  - If you want to add additional objects to an existing file layout, place the cursor on an object of the same type (block, structure, field) and choose *Create*.

- **Delete**

  The *Delete* function is valid for all objects (block, structure, field).

  - Select the object to be deleted and choose *Delete*.

    The objects located below this object in the hierarchy will also be deleted.

    If the object to be deleted (structure, field) is used elsewhere in the file layout, it will NOT be deleted there. If you have deleted all occurrences of the object, the Interface Toolbox will query whether you want to delete the object from the object list. If the object remains in the object list, you can use the object again at any time by choosing *Insert.*

  For an alphabetical list of all structures or fields, see *Structures* or *Fields.*

- **Move**

  The *Move* function is valid for all objects (block, structure, field).

  - Select the object you wish to move.
  - Place the cursor on the object in front of which you wish to insert the object and choose *Move.*

- **Insert**

  The *Insert* function is allowed for structures and fields.

  You can use structures and fields **several times** in the file layout.

- **Attributes**

  The *Attributes* function creates the link with the interface format assigned to the file layout.

User Exits and User-Defined Form Routines

Use

You can use the Interface Toolbox to call your customer form routines during the conversion. This enables you to implement complicated file layouts.

If a user-defined form routine is called up, the Interface Toolbox transfers parameters to the user-defined form routine. The form routine transfers a return value to the Toolbox.

If you want to implement a user-defined form routine using the user exit, you must specify which form routine is used and in which program this form routine can be found.

Prerequisites

The user-defined form routine must meet two criteria:

- It **must** contain **16 parameters**.
- **Parameter 16** is the **return value** for the Interface Toolbox.

If you use a user-defined form routine, you can choose to set the first 15 parameters. These parameters are used as input parameters. If a parameter does not have an entry, then the Interface Toolbox uses SPACE.

You can use the following types of input parameters:

- Constant values as input parameters [Page 134]
- Interface format values as input parameters [Page 135]
- Interface variables as input parameters [Page 136]

Constant Values as Input Parameters

Definition

You can use a constant value as the input parameter for a user-defined form routine. You must enter constant values in quotation marks, for example `"CONSTANT_VALUE".`

The *FIRST* control parameter is used for constant values.

Interface Format Values as Input Parameters

Definition

You can use all data defined in the interface format as input parameters for a user-defined form routine.

Use

This allows you to use data from the export file, in other words, data from internal tables, field strings, and infotypes.

You enter the data as follows: **<table name>-<field name>** (for example, **P0002-NACHN**, **VERSC-ABKRS**, **WPBP-WERKS**).

You must also use the following control parameters for the interface format:

- **First**

  The system uses the value from the first entry.

- **Last**

  The system uses the value from the last entry.

- **Current**

  The system uses the value from the current work area.

You can only use the *Last* and *All* control parameters for internal tables and infotypes. If you use a field string value as the input parameter, you must set the *First* control parameter.

You should only use the **All** control parameter if the user exit is called up within a field or structure assigned to a block. The object is processed according to the data in the repeat factor.

Interface Variables as Input Parameters

Definition

The interface format includes numerous internal interface variables. An interface variable always begins with **&VAR-**.

The *Personnel Number* interface variable could be used as an input parameter as follows: **&VAR-PERNR**.

Use

You must also use the *First* control parameter for the interface variables.

Interface variable	Data type	Length	Meaning
PERNR	Numeric	8	Personnel number from BOP
PERNA	Character	8	Name of export program from BPR
UNAME	Character	12	User name for export program from BPR
DATUM	Date	8	Date of export from BPR
UZEIT	Time	6	Time of export from BPR
VERSN	Numeric	5	Version number from BPR
BEGDA	Date	8	Start date of first payroll period from BOP
ENDDA	Date	8	End date of first payroll period from BOP
PERIO	Numeric	2	Number of exported payroll period from BOP (in-period)
PERBE	Date	8	Start date of exported payroll period from BPE (for-period)
PEREN	Date	8	End date of exported payroll period from BPE (for-period)
RCALC	Character	1	Retroactive accounting indicator from BPE
NPERN	Numeric	6	Number of employees from BPO
VALUE	Character		Value of current structure (dynamic length)
TABNA	Character	10	Current processed table name from BOT
ENTRY	Numeric	6	Number of entries in the current processed table from BOT
NENTR	Numeric	6	Line number of the current entry from BOE
INFBE	Date	8	Start date for current processed infotype from BOI
INFEN	Date	8	End date for current infotype from BOI

WTYEN	Date	8	End date of current processed wage type from BOW
PERMO	Numeric	2	Period modifier from BPE
PABRJ	Numeric	4	Payroll year from BPR
PABRB	Numeric	2	Payroll period BPR
MFILE	Character	40	Output file for conversion (main export file)
AFIL1	Character	40	Additional conversion file 1 (additional export file 1)
AFIL2	Character	40	Additional conversion file 2 (additional export file 2)
AFIL3	Character	40	Additional conversion file 3 (additional export file 3)
AFIL4	Character	40	Additional conversion file 4 (additional export file 4)
APPEN	Character	1	Conversion option *Append File*

Blocks in the File Layout

Use

The blocks form the highest level in the file layout hierarchy. They control the processing of the structures and fields. The interface format stores the objects (block, structure, field) in a graphical overview.

Features

The functions for the blocks are shown as symbols in the tree structure. To access the appropriate function, select an icon.

- *Processing*

  Determines the processing time for a block.

- *Repeat*

  Determines how often a block is processed.

- *Output to file*

  Name of the file to which the export file is written.

- User exit before [Page 139]
- User exit after [Page 140]

Activities

To execute this function you must be in the *Change File Layout <name> (Block View)* screen. Choose *Block, Structure* or *Field.* The blocks are displayed in the selected sequence. The structure and field strings are arranged alphabetically. This is because fields and structures can be used several times.

User Exit Before (Block)

Use

You use the *User Exit Before* function to call a user-defined form routine. The form routine is called **before** the Interface Toolbox starts to process the block, in other words, before the Toolbox processes the structures and fields assigned to a block.

Each user exit **must have 16 parameters**.

Input Values

Parameters 1 to 15 are available as input values for the user exit.

If you use tables, you can select the following parameters for the table entries:

- *First*

  The **first table entry** is transferred to the form routine.

- *Last*

  The **last table entry** is transferred to the form routine.

- *Current*

  The **current table entry** (from the table header) is transferred to the form routine.

Return Value

In the form routine, the return value is parameter 16. The return value of a form routine determines how the block is subsequently processed. The return value can have the following values:

- **0**

  The Toolbox does not process the block.

- **1**

  The Toolbox processes the block.

Activities

In the *User Exit* dialog box, enter the data for the program, for the form routine, and for parameters 1 to 15.

User Exit After (Block)

Use

You use the *User Exit After* function to call a user-defined form routine. The form routine is called **after** the Interface Toolbox has processed all structures and fields belonging to a block. The result for the currently processed block is stored in the local interface block buffer [Page 146], which can be accessed in the user exit.

Each user exit **must have 16 parameters**.

Input Values

Parameters 1 to 15 are available as input values for the user exit.

If you use tables, you can select the following parameters for the table entries:

- *First*

  The **first table entry** is transferred to the form routine.

- *Last*

  The **last table entry** is transferred to the form routine.

- *Current*

  The **current table entry** (from the table header) is transferred to the form routine.

Return Value

In the form routine, the return value is parameter 16. The return value of a form routine determines whether the interface block buffer is written to the export file. The return value can have the following values:

- **0**

  The Interface Toolbox does not write the interface block buffer to the file.

- **1**

  The Interface Toolbox writes the interface block buffer to the file.

Activities

In the *User Exit* dialog box, enter the data for the program, for the form routine, and for parameters 1 to 15.

Structures in the File Layout

Use

A structure corresponds to one line in the export file.

Features

The functions are represented by icons in the object tree. To access a function, select an icon.

- Export

  The export parameters tell the Interface Toolbox the conditions for writing data to the interface block buffer.

- User exit before [Page 142]
- User exit after [Page 143]

User Exit Before (Structure)

Use

You use the *User Exit Before* function to call a user-defined form routine. The form routine is called **before** the Interface Toolbox starts to process the structure, in other words, before the Toolbox processes a particular structure and related fields.

Each user exit **must have 16 parameters**.

Input Values

Parameters 1 to 15 are available as input values for the user exit.

If you use tables, you can select the following parameters for the table entries:

- *First*

  The **first table entry** is transferred to the form routine.

- *Last*

  The **last table entry** is transferred to the form routine.

- *Current*

  The **current table entry** (from the table header) is transferred to the form routine.

Return Value

In the form routine, the return value is parameter 16. The return value of a form routine determines how the structure is subsequently processed. The return value can have the following values:

- **0**

  The Interface Toolbox does not process the structure.

- **1**

  The Interface Toolbox processes the structure.

Activities

In the *User Exit* dialog box, enter the data for the program, for the form routine, and for parameters 1 to 15.

User Exit After (Structure)

Use

You use the *User Exit After* function to call a user-defined form routine. The form routine is called up **after** the Interface Toolbox has processed all fields belonging to a structure. The result for the currently processed block is stored in the local interface block buffer [Page 146], which can be accessed in the user exit.

Each user exit **must have 16 parameters**.

Input Values

Parameters 1 to 15 are available as input values for the user exit.

If you use tables, you can select the following parameters for the table entries:

- *First*

  The **first table entry** is transferred to the form routine.

- *Last*

  The **last table entry** is transferred to the form routine.

- *Current*

  The **current table entry** (from the table header) is transferred to the form routine.

Return Value

In the form routine, the return value is parameter 16. The return value of a form routine determines whether the interface block buffer is written to the export file. The return value can have the following values:

- **0**

  The Toolbox disregards any changes in the form routine and writes the unchanged structure to the interface block buffer.

- **1**

  The Interface Toolbox writes the unchanged structure to the interface block buffer.

Activities

In the *User Exit* dialog box, enter the data for the program, for the form routine, and for parameters 1 to 15.

You can decide whether to transfer the block buffer to the export file in the User Exit After (block).

Field Functions in the File Layout

Use

The fields in the file layout are the lowest level in the object hierarchy. Values are stored in these fields.

Features

The functions are represented by icons in the object tree. To access the appropriate function, select an icon.

- *Length*

  Defines the length of a field.

- *Content*

  Informs the Toolbox how the content of a field is defined.

Calling Specific Interface Data

Use

The Interface Toolbox enables you to select specific data for use in the user-defined form routine.

You can use data from the following areas:

- Interface block buffer [Page 146]
- Interface format data [Page 149]
- Access to export data in a user-defined file layout [Page 151]
- Structure definitions [Page 152]

See also:

Generating the File Layout [Page 156]

Layout Conversion [Page 158]

Interface Block Buffer

Definition

Each file layout, a block buffer is assigned to each block. The Interface Toolbox does not write the block fields to this block buffer at this point. The block buffer is only transferred to the export file when the block has been processed.

Use

If you use the User Exit After [Page 140] function during block processing, you can specify whether the interface block buffer is written to the export file.

The Toolbox defines the interface block buffer and can be used by the user-defined form routine. If your program contains the user-defined form routine and uses the **RPCIFI26** include in the Interface Toolbox, you have unlimited access to the interface block buffer.

In the **RPCIFI26** include, the interface block buffer is defined as an internal table with the name **BLOCKS_OUTPUT**.

```
DATA: BLOCKS_OUTPUT TYPE PINTF_OUTPUT OCCURS 10 WITH HEADER LINE.
TYPES: BEGIN OF PINTF_OUTPUT,
         SNAME(25),
         LENTH(6) TYPE N,
         VALUE TYPE PINTF_MAX_RECORD,
       END OF PINTF_OUTPUT.
TYPES: PINTF_MAX_RECORD (4096) TYPE C.
```

The BLOCKS_OUTPUT-SNAME field contains the name of a structure assigned to a block. The BLOCKS_OUTPUT-LENTH field contains the length of this structure and the BLOCKS_OUTPUT_VALUE field contains the content.

SNAME	LENTH	VALUE
STRUCTURE_01	17	12199601010002X00
STRUCTURE_02	20	Bond, James
STRUCTURE_03	8	19961010

The Interface Toolbox writes the value of the BLOCKS_OUTPUT-VALUE field in the length specified in the BLOCKS_OUTPUT-LENTH field to the output file.

If you use a user-defined form routine that uses the *User Exit After* function for blocks, you can maintain the interface block buffer by deleting, changing, moving, or inserting entries.

```
INCLUDE RPCIFI26.
```

```
FORM MAINTAIN_BLOCK USING PAR_01
                          PAR_02
                          PAR_03
                          PAR_04
                          PAR_05
                          PAR_06
                          PAR_07
                          PAR_08
                          PAR_09
                          PAR_10
                          PAR_11
                          PAR_12
                          PAR_13
                          PAR_14
                          PAR_15     "Input parameters 1 - 15
                       RETURN_VALUE. "return parameter
   RETURN_VALUE = `1`.
LOOP AT BLOCKS_OUTPUT.
    IF BLOCKS_OUTPUT-SNAME = `STRUCTURE_A` AND
       BLOCKS_OUTPUT-VALUE(2) = `XS`.
       CLEAR BLOCKS_OUTPUT-VALUE.
       BLOCKS_OUTPUT-LENTH = 10.
       BLOCKS_OUTPUT-VALUE = `DS19960125`.
       MODIFY BLOCKS_OUTPUT.
   ENDIF.
  ENDLOOP.
ENDFORM.
```

If you change the BLOCKS_OUTPUT interface block buffer and the value **1** is returned to the Toolbox, the Interface Toolbox exports the exact content of the interface block buffer. It is not possible to recreate the previous content of the interface block buffer.

You can use the interface block buffer for each customer form routine that uses the *User Exit After* function for blocks.

Example of an Interface Block Buffer

Interface Block Buffer

Before	SNAME	LENTH	VALUE
	STRUCTURE_A	12	XS0123456789

After	SNAME	LENTH	VALUE
	STRUCTURE_A	10	DS 19960125

Interface Format Data

Use

If you use the *During Processing of Employee Data* processing time to process blocks, structures, and fields in your user-defined form routine, the data can be used as defined in the interface format. The Interface Toolbox generates coding that represents the internal tables, infotypes, and field strings from the interface format.

The coding for the interface format data is defined in the **first include** used to generate the conversion program for the file layout. If you use this include in the program containing the user-defined form routine, the interface format data can be called up in the form <table name>-<field name>. You can also use the standard table operations used in *the Advanced Business Application Programming* (ABAP) language, for example, LOOP, READ TABLE, and so on.

For example, if you have specified that the NACHN, VORNA and GESCH fields from infotype P0002 (*Personal Data*) are used, the Interface Toolbox generates the following coding:

```
DATA: BEGIN OF P0002 OCCURS 5,
        NACHN(000025) TYPE C,
        VORNA(000025) TYPE C,
        GESCH(000001) TYPE C,
END OF P0002.
```

You can now use the names of the fields defined above (NACHN, P0002-VORNA, and P0002-GESCH):

```
REPORT ZUSER_EXITS.
INCLUDE ZPCIFT01.
FORM EXAMPLE_P0002 USING PAR_01
                         PAR_02
                         PAR_03
                         PAR_04
                         PAR_05
                         PAR_06
                         PAR_07
                         PAR_08
                         PAR_09
                         PAR_10
```

Interface Format Data

```
                                PAR_11
                                PAR_12
                                PAR_13
                                PAR_14
                                PAR_15
                             RETURN_VALUE.
READ TABLE P0002 INDEX 1.
CONCATENATE P0002-NACHN
               P0002-VORNA
            INTO RETURN_VALUE
      SEPARATED BY `,`.
ENDFORM.
```

Legend

If the first line of infotype P0002 (*Personal Data*) contains the first name (P0002-VORNA) 'James' and the surname (P0002-NACHN) 'Bond', then the return value is the character string 'Bond, James'.

If the system processes a new payroll period, the Interface Toolbox enters the corresponding payroll period values in the interface format data.

You can also transfer infotype P0002 (*Personal Data*) directly in parameters 1 to 15, instead of following the above example.

Access to Export Data in a User-Defined File Layout

Use

You can set up user exits in different areas (for example, structure, block) within the file layout. If the change validation functions [Page 56] provided do not meet your requirements, you can define customer user exits. In your user exits, you use the function modules included in the standard system to define your change validation function.

Features

You can use the function module within the file layout to enter data for the structures (NEW_ and OLD_ structures) that already exist in the export program for the change validation user exit. SAP recommends that you use this procedure in the following situations:

- You want to access the values for the exported files during change validation, although the system has not identified any changes.
- In the standard system, the change validation function uses the last exported payroll period for comparison. You also want to compare the values for a payroll period with the values for a different period.
- If you have flagged the *Layout conversion directly after export* attribute [Page 52] in the interface format, the tables and field strings (NEW_ and OLD_ structures) still contain values. You can still use the function module to retrieve another comparison period for change validation or to run the conversion of the file layout.

The standard system contains the following function modules:

1. HR_INTF_INITIALIZE

   This function module must be called once before all other function module. You must enter the current program name for the Interface Toolbox.

2. HR_INTF_IMPORT_L

   This function module enters data in the OLD tables in the same way as the *Import* function (IMPRT with specification L).

3. HR_INTF_IMPORT_O

   This function module enters data in the OLD tables in the same way as the *Import* function (IMPRT with specification O).

4. HR_INTF_IMPORT_CURRENT

   This function module enters the current result in the NEW tables.

Structure Definition

Definition

The Interface Toolbox generates a corresponding field string in the second include for each defined structure. This then generates the conversion program for the file layout. You can call each field assigned to a structure, using **<structure name>-<field name>.**

Use

You have defined the STRUCTURE 01 structure. If you have assigned the NAME (constant length: 24) and GENDER (Constant length: 6) fields to the structure, the system generates the following coding:

```
DATA: BEGIN OF STRUCTURE_01,
        NAME(000024) TYPE C,
        GENDER(000006) TYPE C,
END OF STRUCTURE_01.
```

You can use the generated structure information for the User Exit After [Page 143] structure function in the user-defined form routine. If the field content is redefined and the value **1** is returned, the Interface Toolbox will write the structure to the interface block buffer [Page 146] without updating &VAR-VALUE (&VAR-VALUE contains the content of the current structure).

```
REPORT ZUSER_EXITS.
INCLUDE ZPCIFT01.
FORM EXAMPLE_STRUCTURE USING PAR_01
                             PAR_02
                             PAR_03
                             PAR_04
                             PAR_05
                             PAR_06
                             PAR_07
                             PAR_08
                             PAR_09
                             PAR_10
                             PAR_11
                             PAR_12
                             PAR_13
```

```
                              PAR_14
                              PAR_15     "Input parameters 1 - 15
                              RETURN_VALUE.    "Return parameter
   RETURN_VALUE = `1`.
READ TABLE P0002 INDEX 1.
CONCATENATE P0002-NACHN
               P0002-VORNA
               INTO STRUCTURE_01-NAME
      SEPARATED BY `,`.
   IF P0002-GESCH = `1`.
      STRUCTURE_01-GENDER = `MALE`.
   ELSE.
      STRUCTURE_01-GENDER = `FEMALE`.
   ENDIF.
ENDFORM.
```

Legend

If **James** is the value for P0002-VORNA, **Bond** is the value for P0002-NACHN and **1** is the value for P0002-GESCH, then the following entry is made in the BLOCKS_OUTPUT interface format block buffer:

SNAME	LENTH	VALUE
STRUCTURE_01	30	Bond, James MALE

Creating a File Layout

Prerequisites

If you want to access the fields in the interface format [Page 21] when creating the file layout, you must use the following functions for the file layout objects:

- The *Processing* and *Repeat* functions for the *Block* object
- The *Export* function for the *Structure* object
- The *Length* and *Content* of the field for the *Field* object

Procedure

1. Choose the *Configuration* tab index.
2. In the *Object to be processed* field, enter the name of the new file layout.
3. Choose *Create*.

   You access the *Create New File Layout* dialog box.
4. In the *Interface Format* field, enter the name of the interface format whose export file you want to convert. Also enter a short text for the new file layout.
5. Choose *Continue.*

   The *Change File Layout <Name> (Block View)* screen appears.
6. Choose *Create*.

   You access the *Create Block* dialog box.
7. In the *Block Name* field, enter the name of the block and check the default value.
8. Choose *Complete.*

   The object tree for the block is displayed.
9. Select the appropriate function, for example, *Processing* or *Repeat*.

   You access the appropriate dialog box.
10. Make the necessary selection in the dialog box.
11. Place the cursor on the block name and choose *Structure.*

    You access the *Create* dialog box.
12. Choose *Subordinate structure.*

    You access the *Create Structure* dialog box.
13. In the *Structure Name* field, enter the name of the structure.
14. Click the plus sign to the left of the structure name.

    The tree structure for the structure is displayed.
15. Choose the appropriate function, for example, *Export.*

    You access the appropriate dialog box.

16. Make the necessary selection in the dialog box.
17. Place the cursor on the structure name and choose *Fields.*

    You access the *Create Field* dialog box.
18. In the *Field Name* field, enter the name of the field.
19. Click the plus sign to the left of the field name.

    The tree structure for the field is displayed.
20. Select the appropriate function, for example, *Length* or *Contents*.

    You access the appropriate dialog box.
21. Make the necessary selection in the dialog box.
22. Save your entries.

Result:

You have created a customer file layout.

Generating the File Layout

Use

After creating a new file layout or modifying an existing file layout, you must generate the file layout.

If you make changes in the following areas, you must generate the file layout again:

- Interface format [Page 21]
- Import macro for cluster data
- Data definition include [Page 53]
- Change validation [Page 56]
- Wage type processing [Page 86]

Prerequisite

You must enter the following names in the Interface Toolbox before you generate the file layout.

- Name of the file layout program and the include

  You assign the names in the file layout using the File Layout Function program.

- Name of the data definition include

  For more information on file layouts, see File Layout [Page 129].

Activities

Generate the file layout [Page 157]

See also:

Layout Conversion [Page 158]

Generating the File Layout

Prerequisites

You have created a file layout.

Procedure

6. Choose the *Configuration* tab index.
7. In the *File Layout* field, enter the name of the file layout.
8. Choose *Generate.*

Result

The file layout has been generated.

Conversion with the File Layout

Use

Layout conversion converts the export file to another format that has been defined in the file layout. You use this function to run the conversion program.

Features

The following parameters can be used:

- *Input file*

  The name of the file to be converted, in other words the export file created by the export program.

- *Output file*

  The name of the output export file. The converted export file is written to this file.

- *Additional export files 1 to 4*

  The file names used if particular blocks are to be written to an additional file.

- *Append on export files*

  The Toolbox adds the specified file or files to the current conversion output.

  If you do not activate this parameter and the specified file(s) already exist, the system will **overwrite** the existing file(s).

- *Log*

  Detailed information on the conversion procedure.

- *Update*

  If the parameter is flagged, the system writes the output file to the TemSe file.

  If the parameter is not flagged, the system performs a test run.

Prerequisites

Before you can convert the layout, the Interface Toolbox must have successfully exported the data.

For more information on exports, see Export Program [Page 117].

You must also generate the program for the file layout according to the definitions set for blocks, structures, and fields.

For more information, see Generating the File Layout [Page 156].

Activities

Convert the file layout [Page 160]

Converting a File Layout

Prerequisites

1. The Interface Toolbox has successfully exported the data.
2. You have generated the conversion program for the file layout.

Procedure

1. Choose the *Export* tab page.
2. In the *Export activities* group box, select *Conversion with file layout* and enter the name of the file layout.
3. Choose *Execute*.

   The *File Layout <name>* dialog box is displayed.
4. In the *Input and output files* group box, enter the names of the files.
5. In the *Options* group box, enter the name of the required function.

- Append on export files
- Log
- Update

6. Choose *Execute*.

Result

You have used the file layout to convert the file specified under *Import-Export File.* The converted export file is written to the TemSe file (Display TemSe File [Page 126]).

File Format of Export File (SAP Standard)

If an error occurs when the TemSe file is displayed normally [Page 126], the Toolbox displays the export file in an **unformatted** view.

This detailed information on the file format of the export file is required to trace and remove errors, or if you have created a customer program that is based on the SAP standard file layout.

Definition

The export file contains the data that generates the respective export program. The payroll results for each personnel number are grouped together and stored in this file.

The export file is structured so that it has a compact data format and can also be easily read automatically. The compact data format means that majority of data is displayed implicitly, not explicitly. It also ensures that information is saved with minimum redundancy.

Structure of an Export File

Definition

The export file consists of a sequence of bytes that can also be considered to be a sequence of commands in machine language. These commands consist of:

- Operator codes
- Operator parameters (operands)

Structure

Operator Code, Operator Parameter, and Operand

The first two bytes in a command contain the **operator code**. Several bytes follow the operator code and these form the **operator parameter**. The operator parameters represent the **operands**. However, there are also operators without operands.

The BOT operator (**B**egin **o**perator for **t**able) has two operands. The first operand is 10 bytes long, and the second is 6 bytes long. The first operand contains the table name, and the second operand contains the line number of the table entry.

Structure

An export file begins with a start sequence, the **preamble**. The preamble is followed by **personnel numbers**. The **postamble** constitutes the end sequence.

Preamble

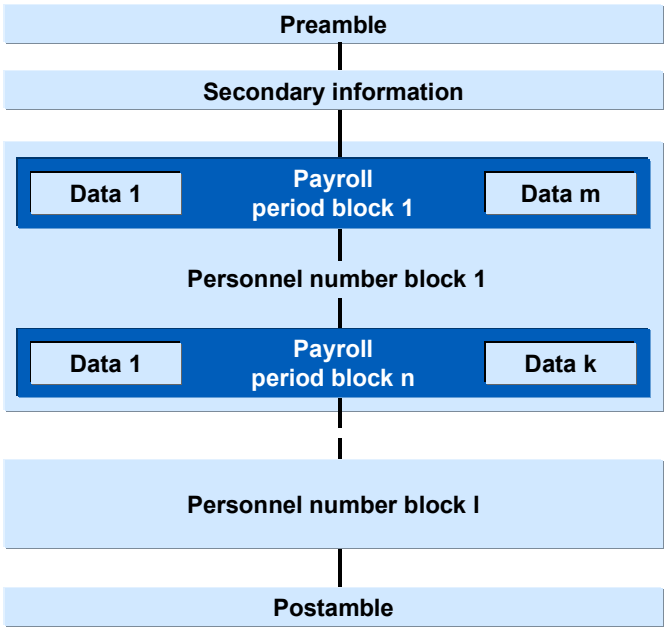
The preamble shows the start of the export file. The operands in the preamble contain information on the export program (for example, creation date). The export file always starts with a preamble.

Personnel number

Each personnel number begins with a **BOP** operator (**b**egin **o**perator for **p**ersonnel number). The payroll period that the **BPE** operator (**b**egin operator **pe**riod) introduces follows. Within the payroll periods there is the actual export data, which is grouped together in blocks. The **EPE** operator (**e**nd operator **pe**rsonnel number) forms the end of the payroll period. The **EOP** operator (**e**nd **o**perator for **p**ersonnel number) is at the end of the personnel number.

Postamble

The postamble is at the end of the export file. The postamble begins with the **BPO** operator (**b**egin operator for **po**stamble), followed by an operand. The **EPO** (**e**nd operator for **po**stamble) operator forms the end of the postamble.



For more information on operators, see Operators for Export/Import Files [Page 165].

See also:

Displaying Export Files Using Operator Blocks [Page 164]

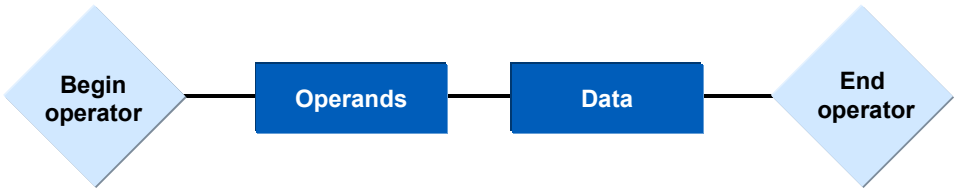
Display Export Files Using Operator Blocks

Definition

Each export file can be displayed as a sequence of **operator blocks**.

Structure

Each block is introduced by a **begin operator** and closed by the accompanying **end operator**. Between the begin and end operators there is a **data section**. This data section can also consist of a sequence of operator blocks.



The **type of operator** gives the operator block the following information:

- Number of operands
- Meaning of the operands
- Structure of the data section
- The structure of the data section is determined by the secondary file.

An export file can consist of the following **operator blocks**:

- Preamble block
- Personnel number block
- Payroll period block
- Postamble block

See also:

Creating the Export File [Page 165]

Operators for the Export File

Definition

The structure of the export file is determined by **operators** and **operands**. A combination of individual operators define the results to be exported. The export file consists of a sequence of operator blocks. Each block begins with a begin operator and ends with an end operator. The type of operator determines how the data between the begin operator and the end operator is interpreted.

Structure

The **operator** is uniquely defined by the **operator code**. The operator code is 12 bytes long and contains a hexadecimal number.

The following operator codes are available:

- Begin of preamble BPR - operator code 01 [Page 167]
- End of preamble EPR - operator code 02 [Page 167]
- Begin of secondary information BSC - operator code 17 [Page 168]
- End of secondary information BSC - operator code 18 [Page 168]
- Begin of personnel number BOP - operator code 05 [Page 169]
- End of personnel number EOP - operator code 06 [Page 169]
- Begin of payroll period BPE - operator code 07 [Page 170]
- End of payroll period EPE - operator code 08 [Page 170]
- Begin of table BOT - operator code 09 [Page 171]
- End of table EOT - operator code 0A [Page 171]
- Begin of table entry BOE - operator code 0B [Page 172]
- End of table entry EOE - operator code 0C [Page 172]
- Begin of field string BOF - operator 0D [Page 173]
- End of field string EOF - operator code 0E [Page 173]
- Begin of infotype BOI - operator code 0F [Page 174]
- End of infotype EOI - operator code 10 [Page 174]
- Begin of wage type BOW - operator code 11 [Page 175]
- End of wage type EOW - operator code 12 [Page 175]
- Begin of postamble BPO - operator code 03 [Page 176]
- End of postamble EPO - operator code 04 [Page 176]

See also:

Structure of the Secondary File [Page 178]

Begin Preamble BPR (01) / End Preamble (02)

Definition

Each export file starts with the preamble block. This block starts with the **BPR** operator, and ends with the **EPR** operator.

The **begin preamble operator** introduces the export file. This operator must be at the beginning of the export file.

No.	Operand	Length	Type	Example
1	Name of export program	40	Character	ZPCIFRX0
2	User name	12	Character	TESTUSER0001
3	Export date	8	Date	19991013
4	Time of export	6	Time	150147
5	Period modifier	2	Numeric	01
6	Payroll year	4	Numeric	1999
7	Payroll period	2	Numeric	01
8	Version number	5	Numeric	00002

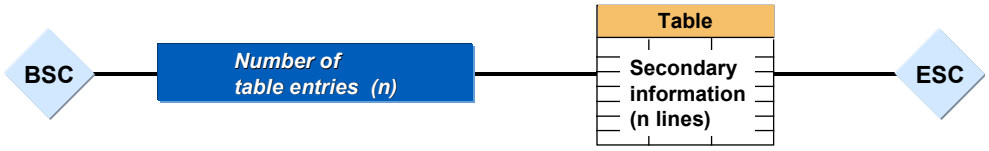
The **end preamble operator** closes the begin preamble operator. This operator has no operands.

Begin of Secondary Information BSC (17)/End of Secondary Information ESC (18)

Definition

Secondary information begins with the **BSC** operator and ends with the **ESC** operator.

Between the BSC operator and the ESC operator, there is a number of table entries and the actual table containing the secondary information.



The **begin secondary information operator** indicates that secondary information follows.

No.	Operand	Length	Type	Example
1	Number of table entries	6	Numeric	000016
2	Table of secondary information	from ABAP Dictionary	PPU12_SEC	

The **end secondary information operator**, which has no operands, closes the begin secondary information operator.

Begin Personnel Number BOP (05) / End Personnel Number EOP (06)

Definition

In the export file, personnel number blocks follow the preamble block. These personnel number blocks start with the **BOP** operator and end with the **EOP** operator.

The **begin personnel number operator** indicates the beginning of a personnel number.

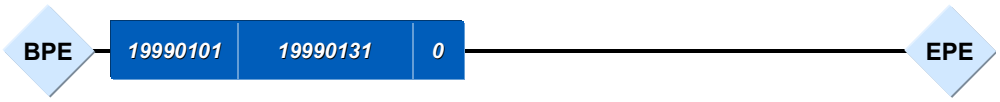
No.	Operand	Length	Type	Example
1	Personnel number	8	Numeric	00900100
2	Start date of first payroll period	8	Date	19990101
3	End date of last payroll period	8	Date	19990331
4	Number of payroll periods	2	Numeric	03

The **end personnel number operator** closes the begin personnel number operator.

Begin Payroll Period BPE (07) / End Payroll Period EPE (08)

Definition

The individual payroll periods in the payroll run are listed within the personnel number blocks. These block begin with the **BPE** operator and end with the **EPE** operator.



The **begin payroll period operator** indicates the beginning of a payroll period.

No.	Operand	Length	Type	Example
1	Start date of payroll period	8	Date	19990101
2	End date of payroll period	8	Date	19990131
3	Retroactive accounting indicator 0=no RA period,1=RA period	1	Character	1
4	Payroll type	1	Character	A: bonus accounting
5	Payroll ID	1	Character	1 to make distinction between several special payroll runs
6	Payroll area	2	Character	A1
7	Period modifier	2	Numeric	01
8	For-period	6	Character	199901
9	Sequence number for cluster IF	9	Numeric	00001
10	Sequence number, payroll result	5	Numeric	00001

The **end payroll period operator** closes the begin payroll period operator. This operator has no operands. It indicates that a payroll period has been processed.

Begin Table BOT (09) / End Table EOT (0A)

Definition

Table data begins with the **BOT** operator and ends with the **EOT** operator.

For each table entry there is an operator pair BOE and EOE between the BOT operator and the EOT operator. The secondary file defines the data between the BOE and EOE.

The second operand of the BOT operator specifies the number of table entries.

The **begin table operator** indicates that table data follows.

No.	Operand	Length	Type	Example
1	Name of table	10	Character	RT
2	Number of table entries	6	Numeric	000016

The **end table operator** closes the begin table operator. This operator has no operands. It indicates that table data has been processed.

Begin of Table Entry BOE (0B) / End of Table Entry EOE (0C)

Definition

For each table entry there is an operator pair BOE and EOE between the BOT operator and the EOT operator. The secondary file defines the data between the BOE and EOE.

The second operand of the BOT operator specifies the number of table entries.

The **begin table entry operator** indicates that table entry data follows.

No.	Operand	Length	Type	Example
1	Line number for entry	6	Numeric	000001

The **end table entry operator**, which has no operands, closes the begin table entry operator. It shows that the table entry data has been processed.

Begin of Field String BOF (0D) / End of Field String EOF (0E)

Definition

The **begin of field string operator** indicates that field string data follows.

No.	Operand	Length	Type	Example
1	Name of field string	10	Character	VERSC

The **end field string operator** closes the begin field string operator. This operator has no operands. It indicates that the field string data has been processed.

Begin of Infotype BOI (0F) / End of Infotype EOI (10)

Definition

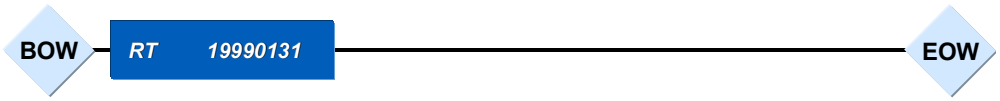
The **begin of infotype operator** indicates that infotype data follows.

No.	Operand	Length	Type	Example
1	Name of infotype	10	Character	P0006
2	Start date for infotype	8	Date	19990101
3	End date for infotype	8	Date	19990131

The **end of infotype operator** closes the begin infotype operator. This operator has no operands. It indicates that infotype data has been processed.

Begin Wage Type BOW (11) / End Wage Type EOW (12)

Definition



The **begin wage type operator** indicates that wage types follow.

No.	Operand	Length	Type	Example
1	Table name for wage type	10	Character	RT
2	End date of wage type	8	Date	19990131

The **end wage type operator** closes the begin wage type operator. This operator has no operands. It indicates that wage type data has been processed.

Begin Postamble BPO (03) / End Postamble EPO (04)

Definition

Each export file ends with the postamble block. This block starts with the **BPO** operator, and ends with the **EPO** operator.

The **begin postamble operator** is almost at the end of the export file. Only the number of employees whose data was exported and the end post operator come after this operator.

No.	Operand	Length	Type	Example
1	Number of employees	6	Numeric	000019

The **end postamble operator** closes the begin postamble operator. This operator has no operands. This is the end of the export file.

Display of Export File - Formatted

The following structure is formatted so it is easier to understand.

```
BPR (01)   ZPCIFX0 TEST_USER_01 19990113 105402 01 1999 01 00002
EPR (02)
  BOP (05)   00900100 19990101 19990131 01
    BPE (07)   19990101 1999 131 1
      BOI (0F)   P0002     19990101 19990131
        NACHN            Bond
        VORNA            James
        GESCH            1
      EOI (10)
      BOT (09)   BT         000001
        BOE (0B)   000001
          BANKL           67290010
          BANKN           123456789
        BETRG           +000500000
        EOE (0C)
      EOT (0A)
      BOW (11)   RT         19990131
        LGART           /101
        BETRG           +000000000500000
      EOW (12)
    EPE (08)
  EOP (06)
BPO (03)   000001
EPO (04)
```

Secondary Files

Definition

The secondary file contains information on each table object (tables, field strings, and infotypes) that you have assigned to the interface format. This is information that can be displayed when you edit the interface format by choosing the *Information* on table object function. The secondary file ensures that the export file can also be processed outside of the SAP System.

Structure

The secondary file, like the export file, can be found in the TemSe file. Each line in the secondary file contains specific information on a table object.

Line Format of Secondary File

No.	Field	Length	Type	Example
1	Name of table	10	Character	RT
2	Field name	10	Character	LGART
3	Data type	1	Character	D
4	Field length	6	Numeric	000008
5	Number of decimal places	6	Numeric	000000
6	Conversion indicator	1	Character	For internal use only
7	Conversion type	2	Numeric	For internal use only
8	Conversion modifier	6	Numeric	For internal use only

Data Types

Data type	Meaning	Example
C	Character	ABCD01
N	Numerical value	038472
D	Date in the form YYYYMMDD	19990101
T	Time in the form HHMMSS	105453
V	Floating point number (with +/- sign)	+0003450
	• 1 byte is the sign (+ or -)	-003746543
	• The last bytes are the decimal places (according to the value in the *Number of decimal places* field.)	

The following example shows a characteristic section of a secondary file. The blank characters are represented by periods.

```
RT........LGART.....C0000040000000000000000
RT........ANZHL.....V0000160000021000000000
RT........BETRG.....V0000160000021000000000
P0002.....NACHN.....C0000250000000000000000
P0002.....VORNA.....C0000250000000000000000
```

See also:

Example: Structure of the Secondary File (Formatted) [Page 180]

Structure of the Secondary File (Formatted)

Field	Data type	Length
P0002-NACHN	C	20
P0002-VORNA	C	20
P0002-GESCH	C	1
BT-BANKL	N	8
BT-BANKN	N	9
BT-BETRG	V	10
RT-LGART	C	4
RT-BETRG	V	16

Generation of Secondary File

Use

The secondary file ensures that the export file can also be processed outside of the SAP System. The secondary file contains all information on the structure of the tables to be used, the field strings, and infotypes.

If you transfer the export file to an third-party payroll system that uses the SAP standard file layout in the Interface Toolbox, you must **regenerate** the secondary file after every change in the interface format.

If you use the import program, you must **regenerate** the secondary file.

If the third-party payroll system uses neither the unconverted export file nor the import program from the Toolbox, then it is **not necessary** to regenerate the secondary file.

Activities

Generate the secondary file [Page 182]

Generating the Secondary File

Prerequisites

You have created the export file for the personnel numbers selected from a payroll area.

Procedure

9. In the menu, choose *File* → *Generate secondary file.*

   You access the *Generate Secondary File* dialog box.

10. In the *Interface Format* and *Secondary File* fields, enter the names of your interface format and secondary file.

    If you leave the *Secondary File* field blank, the secondary file will only be saved in the *Interface Format* (IF) cluster.

    If you enter a name for the secondary file in the *Secondary File* field (using the naming conventions), a secondary file will also be saved in the TemSe file (Displaying the TemSe File [Page 126]).

11. Choose *Display*.
12. Choose *Execute*.
13. Choose *Generate.*

Result

You have generated the secondary file that will be used to further process data from the SAP System.

You can then save the file to a PC by downloading it from the TemSe file.

Import Wage Types

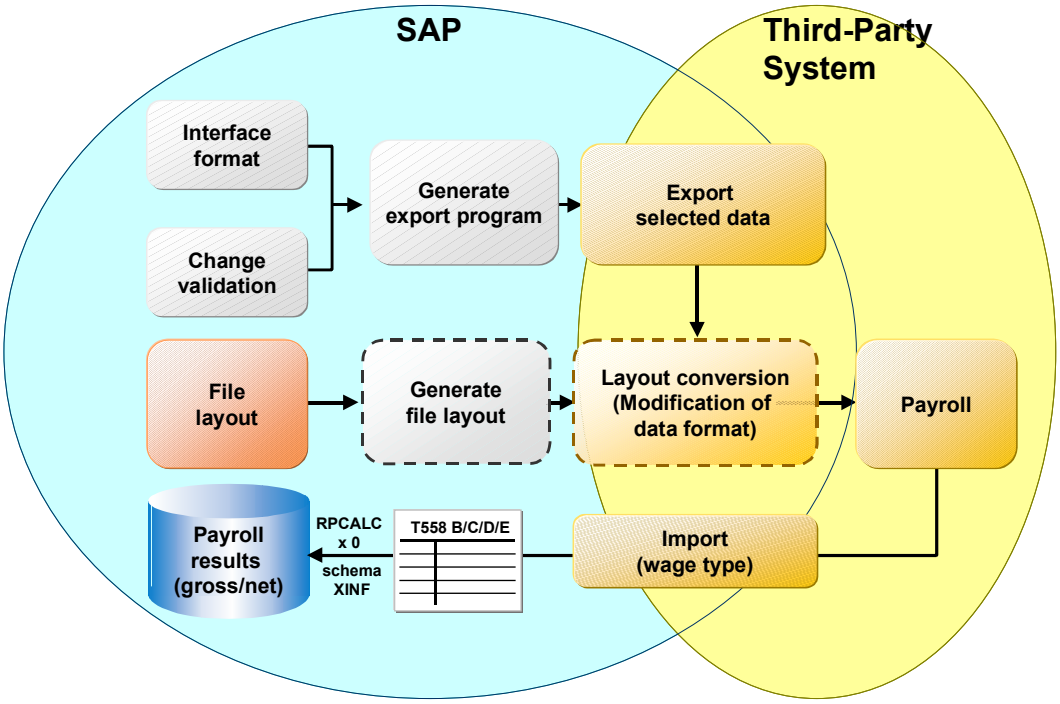
Use

You use the *Import wage types* function to transfer wage types from a third-party system to the payroll results of the SAP System.

For more information, see the Implementation Guide (IMG) for *Cross Application Components* under *Predefined ALE Business Processes → Human Resources → HR External Systems → Connection With an External Payroll System → Import Payroll Results* or in the SAP Library under CA *Cross Application Components → Business Framework Architecture (CA-BFA) → Library of ALE Business Processes → Human Resources → Human Resources - External Applications →* Process Flow: Import Payroll Results from a Third-Party System [Ext.].

Features

The import transfers wage types for payroll from the third-party system to the R/3 System.



Activities

Run the Import [Page 184]

Starting the Import

Prerequisites

You have successfully run payroll in a third-party system.

Procedure

1. In the menu, choose *Import wage types* → *Create IDocs.*

   You access the *IDoc Inbound Processing Via File* screen.

2. In the *Complete file name* field, enter the name of the input file.

3. Choose *Execute*.

   The system message contains the number of created IDocs.

4. In the menu, choose *File* → *Import wage types* → *Create IDocs.*

   You access the *Inbound Processing of IDocs Ready for Transfer* screen.

5. Enter **`MANAGEREXTPAYROLL_INSERTOUT`** as the *Message type* and, if required, enter the *Creation date* and the *Creation time.*

6. Choose *Execute*.

Result

The wage types created in the third-party system are available for further processing in the interface tables of the SAP System.

You can then start payroll in the SAP System and perform the subsequent payroll activities:

- Create a remuneration statement
- Run posting to Accounting