

# Programming Utilities for the Logical Databases PNP and PAP



**Release 4.6C**



## Copyright

© Copyright 2001 SAP AG. All rights reserved.

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

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft<sup>®</sup>, WINDOWS<sup>®</sup>, NT<sup>®</sup>, EXCEL<sup>®</sup>, Word<sup>®</sup>, PowerPoint<sup>®</sup> and SQL Server<sup>®</sup> are registered trademarks of Microsoft Corporation.

IBM<sup>®</sup>, DB2<sup>®</sup>, OS/2<sup>®</sup>, DB2/6000<sup>®</sup>, Parallel Sysplex<sup>®</sup>, MVS/ESA<sup>®</sup>, RS/6000<sup>®</sup>, AIX<sup>®</sup>, S/390<sup>®</sup>, AS/400<sup>®</sup>, OS/390<sup>®</sup>, and OS/400<sup>®</sup> are registered trademarks of IBM Corporation.

ORACLE<sup>®</sup> is a registered trademark of ORACLE Corporation.

INFORMIX<sup>®</sup>-OnLine for SAP and Informix<sup>®</sup> Dynamic Server<sup>™</sup> are registered trademarks of Informix Software Incorporated.

UNIX<sup>®</sup>, X/Open<sup>®</sup>, OSF/1<sup>®</sup>, and Motif<sup>®</sup> are registered trademarks of the Open Group.







HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C<sup>®</sup>, World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA<sup>®</sup> is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT<sup>®</sup> is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

## Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax
	Tip

## Contents

<b>Programming Utilities for the Logical Databases PNP and PAP .....</b>	<b>5</b>
Definition of Constants .....	6
Skipping Employees Who Have Left the Company .....	7
Definition of the Selection Period .....	8
Definition of the Data Selection Period .....	9
Name Formatting in HR Reports .....	10
Formatting Names in Accordance with National Regulations .....	11
HR-Formatting Address in Accordance with Mailing Regulations.....	13
Check and Output of Personnel Identification Numbers.....	14
Output of List Period .....	16
Output of List Key Date.....	17
Initialization of Buffer for Accessing PCLn.....	18
Import/Export Macros for Cluster Data .....	19
Import Macros .....	21
Export Macros .....	23
Deleting Payroll Results from the Memory .....	24
Read the Infotype for One Person in a Period.....	25
Provide the First Entry in the Period .....	27
Provide the Last Entry in the Period.....	29
Changing HR Data .....	31

## Programming Utilities for the Logical Databases PNP and PAP

### Use

Programming utilities enable you to integrate similar requirements in programs. You can use this module for your own reports.

The following programming utilities are available for the logical databases PNP and PAP:

- Function modules:
- Macros
  - Macros have an advantage in that each report using this macro is automatically regenerated when called if the macro changes.
- RMAC macros
  - RMAC macros represent specific macros used in the *Human Resources* component (HR). However, they do have a disadvantage in that a change to a RMAC macro must be repeated manually in all reports using the RMAC macro.



We recommend that you use macros or function modules in your reports, provided they can replace the RMAC macros.

### Features

Programming utilities are available at the following events in the program flow:

- Data definition
- At INITIALIZATION
- At START-OF-SELECTION
- At SELECTION (GET PERNR)
- At TOP-OF-PAGE
- At any point
  - (for example, SELECTION, END-OF-SELECTION, AT PFxx, ...)
  - Retrieving data
  - Processing data
  - Changing data

---

**Definition of Constants**

## Definition of Constants

### Use

You can enter the parameters for the data definition of the earliest date in the system (LOW-DATE) and the latest date (HIGH-DATE) directly in the program. You can do this as follows:

- *CONSTANTS: LOW-DATE TYPE D VALUE '18000101'*
- *CONSTANTS: HIGH-DATE TYPE D VALUE '99991231'*



They represent the earliest and latest points on the time line and are not actual points in time. Note that you can only use these dates for comparison, not for calculation.

## Skipping Employees Who Have Left the Company

### Use

If employees have left the company and you do not want to include them in the data selection, you can use the following programming utility:

RMAC macro: RP-SEL-EIN-AUS-INIT

You use this RMAC macro at the INITIALIZATION event.

Using the RMAC macro *RP-SEL-EIN-AUS-INIT* gives you the following advantages:

- *RP-SEL-EIN-AUS-INIT* suggests a default value that the report user can change on the selection screen at the start of the report.
- *RP-SEL-EIN-AUS-INIT* allows relatively early processing by the database processor.
- *RP-SEL-EIN-AUS-INIT* refers to the interval of time (PN-BEGPS,PN-ENDPS).
- The selection of database processor initiated by *RP-SEL-EIN-AUS-INIT* is 'simultaneously' linked to the other database selection criteria.

### Features

If you use the RMAC macro *RP-SEL-EIN-AUS-INIT*, the system suggests that you only select employees who have not left the company when the report selection screen is displayed. The *Status - employment* field automatically contains a default value which means that only employees who have left the company are selected.

### Parameters

None

### Check

None

### Prerequisite

None

### Example

```
...  
INITIALIZATION.  
RP-SEL-EIN-AUS-INIT.  
START-OF-SELECTION.
```

---

**Definition of the Selection Period**

## Definition of the Selection Period

### Use

With the global variable *PNPTIMED*, you can set the selection period for the selection screen at the *INITIALIZATION* event. For example, you can create your program so that the system suggests the values for an evaluation up to the current day when the report is called.

### Features

The possible specifications of variable *PNPTIMED* are:

- D for *key date*
- M for *month*
- Y for *year*
- P for *past*
- F for *future*

### Example

```
INITIALIZATION.
```

```
...
```

```
PNPTIMED = D
```



The ABAP statement means that the system suggests a key date evaluation on the report selection screen when the report is called.



## Definition of the Data Selection Period

### Use

The following programming utilities are available for changing the data selection periods for individual infotypes:

Macro: *RP\_SET\_DATA\_INTERVAL*

You use the macro at the INITIALIZATION event.

### Features

With the macro *RP\_SET\_DATA\_INTERVAL*, you can change the data selection period for individual infotypes at the INITIALIZATION event.

### Parameters

IN	1)	Infotype
	2)	beg Start of the time interval
	3)	end Validity end date of the time interval
OUT		None

### Check

None

### Example

```
INITIALIZATION.
  RP_SET_DATA_INTERVAL 0001 SY_DATUM SY_DATUM
START_OF_SELECTION.
```

## Name Formatting in HR Reports

## Name Formatting in HR Reports

### Use

The following programming utilities are available for formatting the name in HR reports:

Function module: *RP\_SET\_NAME\_FORMAT*

HR name formatting is done using the function module RP-EDIT-NAME at selection (GET PERNR). This routine edits the name according to entries in the table T522N (HR Name Editing). As HR name formatting can be dependent on the program, the function module *RP\_SET\_NAME\_FORMAT* is directly linked to HR name formatting in the START-OF-SELECTION event.



You should use this function module macro in general HR reports where a list of employee names is created. (for example, *Employee List* (RPLMIT00), *Time Spent in Each Pay Scale Area/Type/Group/Level* (RPLTRF00), *Defaults for Pay Scale Reclassification* (RPLTRF10)).

### Features

The function module *RP\_SET\_NAME\_FORMAT* determines the respective report format from the *Format for HR Name Editing* table (T522F). If no entry has been made in this table for the corresponding HR report, the report format is set to 01 (DEFAULT).

### Parameters:

IN :	Report name
OUT:	Report format from T522F (default: '01') for T522N

### Check:

Table T522F (*Format for HR Name Editing*)

## Formatting Names in Accordance with National Regulations

### Use

The following programming utilities are available for formatting the name in HR reports:

Function Module: *RP\_EDIT\_NAME*



You should use the function module *RP\_EDIT\_NAME* in all HR reports in which the employee's complete name is printed.

You use the function module at the SELECTION event.

### Features

The function module *RP\_EDIT\_NAME* formats names in all HR standard evaluations in compliance with the national regulations of a particular country.

Names are formatted according to the entries in table T522N (HR name editing).

HR name formatting is also dependent on:

- MOLGA (company code, personnel area, personnel subarea) from the *Personnel area/Personnel subarea* view (V\_T001P)
- The report format from the *Format for HR Name Format* table (T522F) (Default: '01')
- Origin of the data
  - *Personal Data* infotype (0002)
  - *Family/Related Person* infotype (0021)
  - *Family Data* infotype (0148)
- Name formatting indicator from the *Personal Data* infotype (0002) (field *Special form* (KNZNM))

### Parameters

IN	1)	Infotype P0001	Org. assignment (P0001-ENAME)
	2)	Infotype P0002	Personal data
		Infotype P0021	Family/related person
		Infotype P0148	Family data (Japan)
	3)	MOLGA	(T001P-MOLGA)
	4)	LANGU	Language for form of address
			SPACE SAP logon language SY-LANGU
			x Receiver language P0002-SPRSL
	5)	\$\$FORMAT	Name format for formatting

**Formatting Names in Accordance with National Regulations**

	6)	NAMEL	Maximum length of EDIT-NAME
OUT	1)	\$EDIT-NAME	Formatted name string for transfer
	2)	\$RET-CODE	Return code for name formatting

**Check**

- Table T522F (*Format for HR Name Editing*)
- Table T522N (*HR Name Editing*)

## HR-Formatting Address in Accordance with Mailing Regulations

### Use

To format addresses in accordance with postal regulations, use the following program utility:

Function Module: *HR\_MAKE\_ADDRESS*

You use the function module at the SELECTION event.

### Features

The function module *HR\_MAKE\_ADDRESS* calls the function module *ADDRESS\_INTO\_PRINTFORM*. This formats the employee's address in accordance with international postal conventions.

For information on how the individual addresses are processed, see the documentation on the function module *ADDRESS\_INTO\_PRINTFORM*.

### Parameters

IN	1) Infotype P0001	Org. Assignment (P0001-BUKRS for T001F)
	2) Infotype P0002	Personal Data (P0002-ANRED for form of address)
	3) Infotype P0006	Address data
	4) P0001-ENAME	Name in compliance with national regulations (use with RP-EDIT-NAME)
	5) LINE_COUNT	Number of printed lines
	6) Name	RP_EDIT_NAME or CNAME from the <i>Personal Data</i> infotype (0002)
OUT	ADRS	Structure of formatted address

### Check

None

Check and Output of Personnel Identification Numbers

## Check and Output of Personnel Identification Numbers

### Use

Personnel identification numbers are mainly used by the HR country versions (for example, USA, Canada, Denmark). For example, the social insurance or tax number for an employee can be entered in the *Personal Data* infotype (0002) as a personnel identification number.

In some reports, it can be useful to take these personnel identification numbers into account.

The following programming utilities are available to check and print personnel identification numbers from the *Personal Data* infotype (0002):

- Function module: *RP\_FETCH\_ALTERNATE\_PERNR*
- RMAC macro: *RP-WRITE-ALTER-PERID*



You should use these programming utilities in general HR reports (for example, *Employee List* (RPLMIT00), *Time Spent in Each Pay Scale Area/Type/Group/Level* (RPLTRF00), *Defaults for Pay Scale Reclassification* (RPLTRF10)).

You use the programming utilities at the START-OF-SELECTION event.

### Features

The system uses the function module *RP\_FETCH\_ALTERNATE\_PERID* to check the personnel identification numbers from the *Personal Data* infotype (0002).

#### Parameter *RP\_FETCH\_ALTERNATE\_PERNR*

OUT:	1) SW-ALTER-PERID 2) RETCODE	Flag alternative PERID (YES/NO) Return code from RE549B
------	---------------------------------	--

### Check

- Transaction: *HR: Features* (PE03)
- Feature: *Consideration of Personnel ID from Infotype 0002* (PERNO)
- The function module uses the country grouping for the user (T513A) to determine whether a personnel identification number exists.



If you also use the RMAC macro *RP-WRITE-ALTER-PERID*, the personnel identification numbers are checked and displayed by the system.

The RMAC macro *RP-WRITE-ALTER-PERID* makes sure that personnel identification numbers are printed in a separate output line in the report output list.

#### Parameter *RP-WRITE-ALTER-PERID*

IN	1) \$\$SW-ALTER-PERID	Switch alternative PERID (YES/NO)
	2) P0002	<i>Personal Data</i> infotype (0002)

**Check and Output of Personnel Identification Numbers**

	3) Print option	⌘ direct in RMAC module or SPACE indirect in user report
OUT	1) PERID	Personnel identification P0002
	2) \$LENGTH	Length of personnel identification

**Check**

- Transaction: *HR: Features* (PE03)
- Feature: *Consideration of Personnel ID from Infotype 0002 (PERNO)*
- Var. key: User value from *User-Related Values* table (T513A) (MOLGA)

---

**Output of List Period**

## Output of List Period

### Use

If you want to display the period in the output list of an HR report, use the following program utilities:

RMAC macro: *RP-ZEITRAUM*

You use this RMAC macro at the TOP-OF-PAGE event.

### Features

The RMAC macro *RP-ZEITRAUM* displays the period (PN-BEGDA, PN-ENDDA) on the top line of each page of each page in the standard HR list.

### Parameters

None

### Check

None

### Example

```
...  
TOP-OF-PAGE.  
IF PN-BEGDA EQ PN-ENDDA.  
  RP-STICHTAG.  
ELSE.  
  RP-ZEITRAUM.  
ENDIF.
```



## Output of List Key Date

### Use

If you want to display the key date in the output list of an HR report, use the following program utilities:

RMAC macro: *RP-STICHTAG*

You use this RMAC macro at the TOP-OF-PAGE event.

### Features

If you use the RMAC module *RP-STICHTAG*, the system prints the key date (PN-ENDDA) in the first line of each page in the list.

### Parameters

None

### Check

None

### Example

```
(RP-STICHTAG)
...
TOP-OF-PAGE.
IF PN-BEGDA EQ PN-ENDDA.
  RP-STICHTAG.
ELSE.
  RP-ZEITRAUM.
ENDIF.
```

## Initialization of Buffer for Accessing PCLn

# Initialization of Buffer for Accessing PCLn

## Use

Initializes the main memory buffer for accessing files PCLn (n = 1, 2, 3, 4) and is used together with the [import \[Page 21\]](#) macros and [export \[Page 23\]](#) macros.

Macro: *RP-INIT-Buffer*



For detailed information on importing and exporting files, see [Import/Export Files in HR \[Ext.\]](#).

## Parameters

None

## Check

None

## Integration

The following includes contain the data definition for the buffer. They must be included in the report that writes the data to or reads the data from the database.

- RPPPXD00x
- RPPPXD10
- RPPPXM00



See the example [Starting Payroll in the Test Mode \[Ext.\]](#).

## Import/Export Macros for Cluster Data

### Definition

Programs that process the cluster data (for example, RX) do not access the cluster independently. The data is accessed using a defined interface created with macros.

The import and export macros form the basis for the structure of a cluster. The structure definition for a cluster contains a list of the stored data and the sequence in which it is used.



For more information on the structure description, see [Files PCL1, PCL2, PCL3, and PCL4 \[Ext.\]](#).

### Use

By using macros, you ensure that the cluster data is always accessed under the same conditions. This avoids inconsistencies.

If there are changes in the way the data is read or stored, then by using macros, these changes must only be made in one place (in the respective macro).



The macro coding must be available in the program. Macros are inserted in the reports using includes.

### Structure

The following **naming convention** applies:

RP-aaa-bb-cc

Legend:

- *aaa* is the type of macro:
  - IMP** Import macro
  - EXP** Export macro
- *bb* is the database table where the data is saved:
  - C1** Database object PCL1
  - C2** Database object PCL2
  - C3** Database object PCL3
  - C4** Database object PCL4
- *cc* is the cluster:
  - RX** Cluster object RX
  - RD** Cluster object RD
  - B2** Cluster object B2and so on

**Import/Export Macros for Cluster Data**

Macros are defined using the ABAP commands DEFINE/END-OF-DEFINITION.

## Import Macros

### Use

Reads field, field strings or internal tables from the database.

Macro: *RP-IMP-Cn-xx*

The macros use the ABAP command `IMPORT` to manage the PCLn database tables. The PCLn are read with the unique key [xx-Key \[Ext.\]](#).



Only use the macros *RP-IMP-Cn-xx* or the function module `PYXX_READ_PAYROLL_RESULTS` to import the data.

When the macros are used to import data, the data records are not read directly from file PCLn. Instead, the system checks the buffer directory to see whether the main memory already contains a record with the same key. If this is not the case, the record is read from PCLn to the buffer and then retrieved from the buffer for the report.

If the import is successful, the return code `RP-IMP-xy-SUBRC = 0` is set. When data is read from the buffer, the system carries out a check for cluster authorization. Standard import programs follow the naming convention `RPCLSTxy` (xy = cluster name).



The main import macros *RP-IMP-C2-Rx* for the payroll cluster Rx can be found in the includes `RPCXRxx0` (for example, *RP-IMP-C2-RU* in include `RPCXRUU0`).

### Parameters

None

The appropriate [xx-Key \[Ext.\]](#) must, however, be specified before the module is accessed.

### Check

None

### Prerequisite

Depending on the dataset that has already been read, the main memory buffer must have been initialized with the macro [RP-INIT-BUFFER \[Page 18\]](#).



For information on importing and exporting files, see [Import/Export Files in HR \[Ext.\]](#).

## Example

An import macro that reads data from cluster TT may have the following structure:



```
DEFINE RP-IMP-C2-TT.  
IMPORT TABLE_01
```

**Import Macros**

```
FIELDSTRING_01
TABLE_02
TABLE_03
FROM DATABASE PCL2 (TT) ID TT-KEY.
END-OF-DEFINITION.
```

In this example, the following objects are read from cluster TT of the database table PCL2:

- Internal tables TABLE\_01, TABLE\_02, TABLE\_03
- Field string FIELDSTRING\_01

## Export Macros

### Use

Stores field, field strings or internal tables on the database.

Macro: *RP-EXP-Cn-xx*

The macros use the ABAP command `EXPORT` to manage the PCLn database tables. The assignment takes place with the unique key [xx-Key \[Ext.\]](#).



Only use the macros *RP-EXP-Cn-xx* or the function module `PYXX_WRITE_PAYROLL_RESULTS` to export the data.

When macros are used for exporting, records are written to a main memory buffer and not directly to the database. When the program run has been completed, the records in the buffer are stored in the appropriate PCLn database.

Saving the records stored in the buffer can also be forced using the routine `Update`.



The main export macros *RP-EXP-C2-Rx* for the payroll cluster *Rx* can be found in the includes *RPCnRxx0* (for example, *RP-EXP-C2-RU* in include *RPC2RUU0*).

### Parameters

None

The appropriate [xx-Key \[Ext.\]](#) must, however, be specified before the module is accessed.

### Check

None

### Prerequisite

Depending on the dataset that has already been read, the main memory buffer must have been initialized with the macro [RP-INIT-BUFFER \[Page 18\]](#).



Module *RP-EXP-C2-RA* writes the payroll results back from the memory to the main memory buffer (file *PCL2*, cluster *RA*).



For information on importing and exporting files, see [Import/Export Files in HR \[Ext.\]](#).

---

**Deleting Payroll Results from the Memory**

## Deleting Payroll Results from the Memory

### Use

This function deletes data stored in the memory for cluster xx, file PCLn. The structure is retained in the memory.

Macro: *RP-REF-Cn-xx*



The data is not deleted from the main memory buffer or the database.

### Parameters

This module has no parameters. The appropriate [xx-Key \[Ext.\]](#) must, however, be specified before the module is called.

### Check:

None



Module RP-REF-C2-RA deletes the payroll results stored in the memory for cluster RA, file PCL2.



## Read the Infotype for One Person in a Period

### Use

To place all infotype data records for an employee for a specified period in an internal table, use the following programming utility:

Macro: *RP\_READ\_INFOTYPE*

You define the macro using the keyword `INFOTYPES`.

You can use the macro in all programs at any point. You can also use it in function modules.



In database PNP, an infotype is usually read with `GET PERNR`. Using macro *RP\_READ\_INFOTYPE* is an exception.

You can also use the function module *HR\_READ\_INFOTYPE*. For information on how to use the function module, see the documentation on Function Modules.

### Prerequisites

- The validity begin date of the time period must be before or the same as the validity end date.
- Validity begin and end are correct date specifications (preferably of the type DATE).
- The infotype table must match the infotype number.
- The program using the macro must contain the include `DBPNPMAC`.

### Features

The macro *RP\_READ\_INFOTYPE* makes sure that all data records for a person for the specified period are placed in an internal infotype table.

### Parameters

`RP_READ_INFOTYPE pernr infty inftytab beg end`

IN :	1)	Personnel number of the person requested
	2)	Infotype number of the required infotype
	3)	Name of the internal infotype table
	4)	Validity start date of the time interval
	5)	Validity end date of the time interval
OUT:	1)	PNP-SW-FOUND = 0, if there is no matching record in the dataset
		PNP-SW-FOUND = 1, if there is no matching record in the dataset
	2)	PNP-SW-AUTH-SKIPPED-RECORD = 0, if the HR authorization check has not retained any records due to incorrect authorizations.

**Read the Infotype for One Person in a Period**

		PNP-SW-AUTH-SKIPPED-RECORD = 1, if the HR authorization check has retained at least one record due to lack of authorization
	3)	Internal infotype table, containing all matching records for which the user is authorized (this table can also be empty).

**Check**

None

**Example**

```

(RP_READ_INFOTYPE pernr infty inftytab beg end)
  INFOTYPES: 0001.
  RP-LOWDATE-HIGHDATE.
  DATA: PERNR LIKE P0001-PERNR.
  DATA: BEGDA LIKE P0001-BEGDA, ENDDA LIKE P0001-ENDDA.
  PERNR = '12345678'.
  BEGDA = LOW-DATE + 15
  ENDDA = HIGH-DATE - 5.
  RP-READ-INFOTYPE PERNR 0001 P0001 BEGDA ENDDA.
  IF PNP-SW-AUT-SKIPPED-RECORD EQ '1'.
  WRITE: / 'Insufficient authorization'. STOP.
  ENDIF.
  IF PNP-SW-FOUND EQ '0'.
  WRITE: / 'Infotype 0001 missing'. STOP.
  ENDIF.

```

## Provide the First Entry in the Period

### Use

Use the following programming utility to place the first entry in a required period (this can be a for a subtype) in the table header entry from an internal infotype table.

Macro: *RP\_PROVIDE\_FROM\_FRST*

You define the macro using the keyword `INFOTYPES`.

You use macro *RP\_PROVIDE\_FROM\_FRST* in programs for the logical databases PNP and PAP where the first data record for a period (can be a subtype) is read from an infotype table. The infotype table has been filled earlier (for example, with `GET PERNR` or *RP\_READ\_INFOTYPE*). This macro is only helpful if the infotype has time constraint 1 or 2.

### Prerequisites

- The validity begin date of the time period must be before or the same as the validity end date.
- Validity start and end dates are correct (preferably of the type `DATE`).
- The infotype table is sorted in ascending order. Otherwise, you would receive the first fitting table entry that might not necessarily correspond to the first time entry.

### Features

The first entry for a specified period is placed in the table header entry from an internal infotype table.

### Parameters

`RP_PROVIDE_FROM_FRST` infitytab subty beg end

IN :	1)	Name of the internal table
	2)	Subtype required or <code>SPACE</code> if no subtype is being specified
	3)	Validity start date of the time interval
	4)	Validity end date of the time interval
OUT:	1)	PNP-SW-FOUND: has the value 0 if there is no matching entry in the infotype table in the given time period. Otherwise it has the value 1.
	2)	The matching table header entry if <code>PNP-SW-FOUND = 1</code> or the initial table header entry if <code>PNP-SW-FOUND = 0</code>

### Check

None

---

**Provide the First Entry in the Period****Example**

```
(RP_PROVIDE_FROM_FRST inftytab subty beg end)
RP_PROVIDE_FROM_FIRST P0021 '1' PN-BEGDA PN-ENDDA.
IF PNP-SW-FOUND EQ '1'.
...
or
RP_PROVIDE_FROM_FRST P0001 SPACE PN-BEGDA PN-ENDDA.
IF PNP-SW-FOUND EQ '0'.
WRITE: / 'Error: Org. assignment is missing' REJECT.
ENDIF.
```

## Provide the Last Entry in the Period

### Use

Use the following programming utility to place the last entry in a required period (this can be a for a subtype) in the table header entry from an internal infotype table.

Macro: *RP\_PROVIDE\_FROM\_LAST*

You define the macro using the keyword *INFOTYPES*.

You use macro *RP\_PROVIDE\_FROM\_LAST* in programs for the logical databases PNP and PAP where the last data record for a period (can be a subtype) is read from an infotype table. The infotype table has been filled earlier (for example, with *GET PERNR* or *RP\_READ\_INFOTYPE*). This macro is only helpful if the infotype (or subtype) has time constraint 1 or 2.

### Prerequisites

- The validity begin date of the time period must be before or the same as the validity end date.
- Validity start and end dates are correct (preferably of the type *DATE*).
- The infotype table is sorted in ascending order. Otherwise, you would receive the last fitting table entry that might not necessarily correspond to the last time entry.

### Features

The macro *RP\_PROVIDE\_FROM\_LAST* makes sure that the last entry for a specified period is placed in the table header entry of the report output list.

### Parameters

*RP\_PROVIDE\_FROM\_LAST* infitytab subty beg end

IN :	1) Name of the internal table
	2) Subtype required or <i>SPACE</i> if no subtype is being specified
	3) Validity begin date of the time interval
	4) Validity end date of the time interval
OUT:	1) PNP-SW-FOUND: has the value 0 if there is no matching entry in the infotype table in the given time period. Otherwise it has the value 1.
	2) The matching table header entry if PNP-SW-FOUND = 1 or the cleared table header entry if PNP-SW-FOUND = 0

### Check

None

---

**Provide the Last Entry in the Period****Example**

```
(RP_PROVIDE_FROM_LAST inftytab subty beg end)
RP_PROVIDE_FROM_LAST P0021 '1' PN-BEGDA PN-ENDDA.
IF PNP-SW-FOUND EQ '1'.
...
or
RP_PROVIDE_FROM_LAST P0001 SPACE PN-BEGDA PN-ENDDA.
IF PNP-SW-FOUND EQ '0'.
WRITE: / 'Error: Org. assignment is missing'. REJECT.
ENDIF.
```



The module PROVIDE-FROM-FINAL, which is not implemented, is a special case of PROVIDE-FROM-LAST:

```
PROVIDE-FROM-FINAL inftytab subty beg end   =
  RP_PROVIDE_FROM_LAST inftytab subty end end
```

## Changing HR Data

### Use

If you want to change HR data, use the following programming utilities:

Macro: *RP\_UPDATE*

You use the macro as an interface between reports that make database changes for the infotype table. You only use the macro to change data records. You cannot add or delete data records.

You only use the macro in programs that change data. You can use the macro at any point in a program.

### Prerequisites

- All of the data records you want to change are also included in the table in their original form.
- No key changes, that is, no adding or deleting data.

### Features

#### Parameters

RP-UPDATE old-table new-table

IN :	1) Name of internal table with infotype records before change.
	2) Name of internal table with infotype records after change.

### Check

None

### Example

RP-UPDATE OLD-P0003 P0003.

Filename: EXP\_OOOO.doc  
Directory: C:\TEMP\iwbprint0001  
Template: C:\Users\templates\sapkeeh.dot  
Title: Programming Utilities for the Logical Databases PNP and PAP  
Subject:  
Author: SAP AG  
Keywords:  
Comments: Release 46C 23.07.2001  
Creation Date: 23.07.01 09:40  
Change Number: 2  
Last Saved On: 23.07.01 09:40  
Last Saved By: SAP AG  
Last Printed On: 23.07.01 10:13  
As of Last Complete Printing  
Number of Pages: 31  
Number of Words: 4.628 (approx.)  
Number of Characters: 24.532 (approx.)