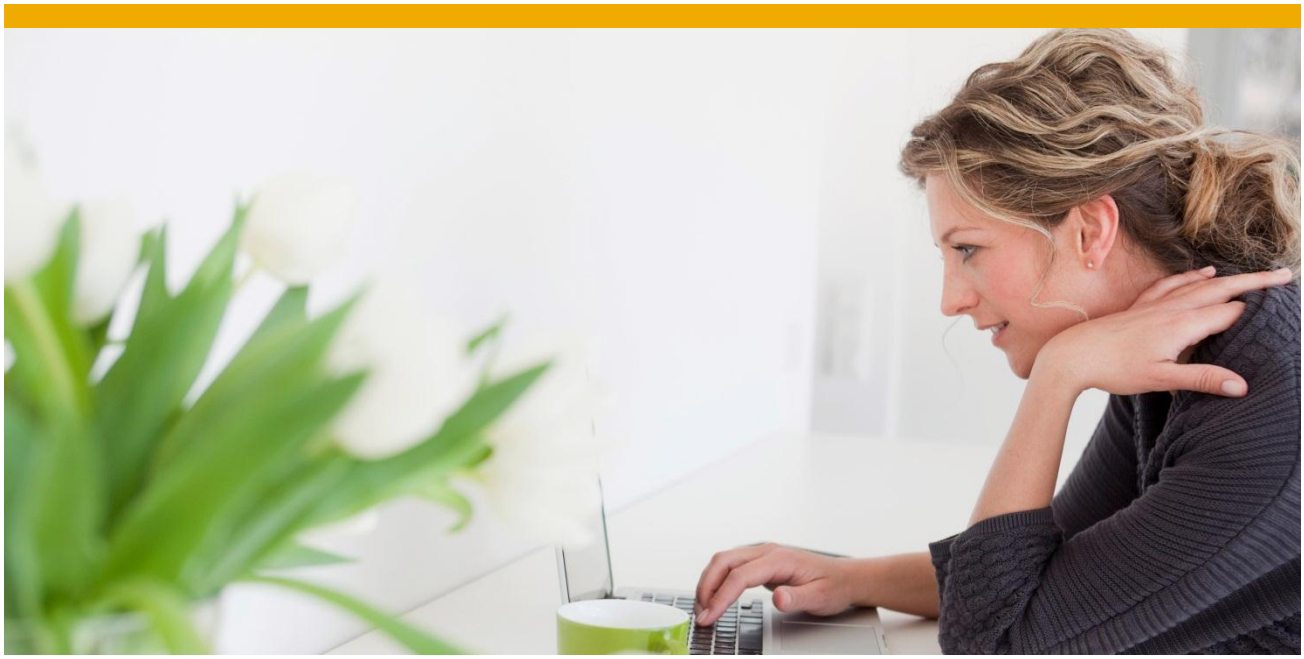


Using Purchase Orders in the NW Enterprise Procurement Model

The programming model of the NetWeaver Enterprise Procurement Model (EPM) follows the Business Object (BO) paradigm. A BO encapsulates the business logic of a business entity, such us a Purchase Order. In this document, we show how you to handle EPM Business Objects like this from a programmatic point of view. After reading this document and looking at the sample code, you will be ready to create your own EPM BO-based code.

General information about EPM is available in The NetWeaver Enterprise Procurement Model - An Introduction on the SAP Community Network.

TABLE OF CONTENTS

PURCHASE ORDER BO AND RELATED BOS

A Business Object implements the business logic of an EPM business entity, such as a Purchase Order BO. Business Objects allow service-oriented access to EPM business entities. You can use a Purchase Order BO for example to create a new Purchase Order by requesting the corresponding service from the BO.

Implementation aspects

This document shows how to

- Create a Purchase Order with Items and Schedule Lines
- Assign a Business Partner to the Purchase Order Header
- Assign a Product to a Purchase Order Item
- Create a Goods Receipt Header and Goods Receipt Items relating to the Purchase Order
- Create a Purchase Order Invoice and Purchase Order Items relating to the Purchase Order
- Set the Purchase Order status with the corresponding action methods
- Save the data of all created objects
- Read the Purchase Order data and related objects from the storage
- Retrieve data using query methods

The sample focuses on back-end implementation. It does not show how to use the API parts for user interfaces.

To invoke an action method of a PO, it must have a certain status. This status and action model is described in the appendix.

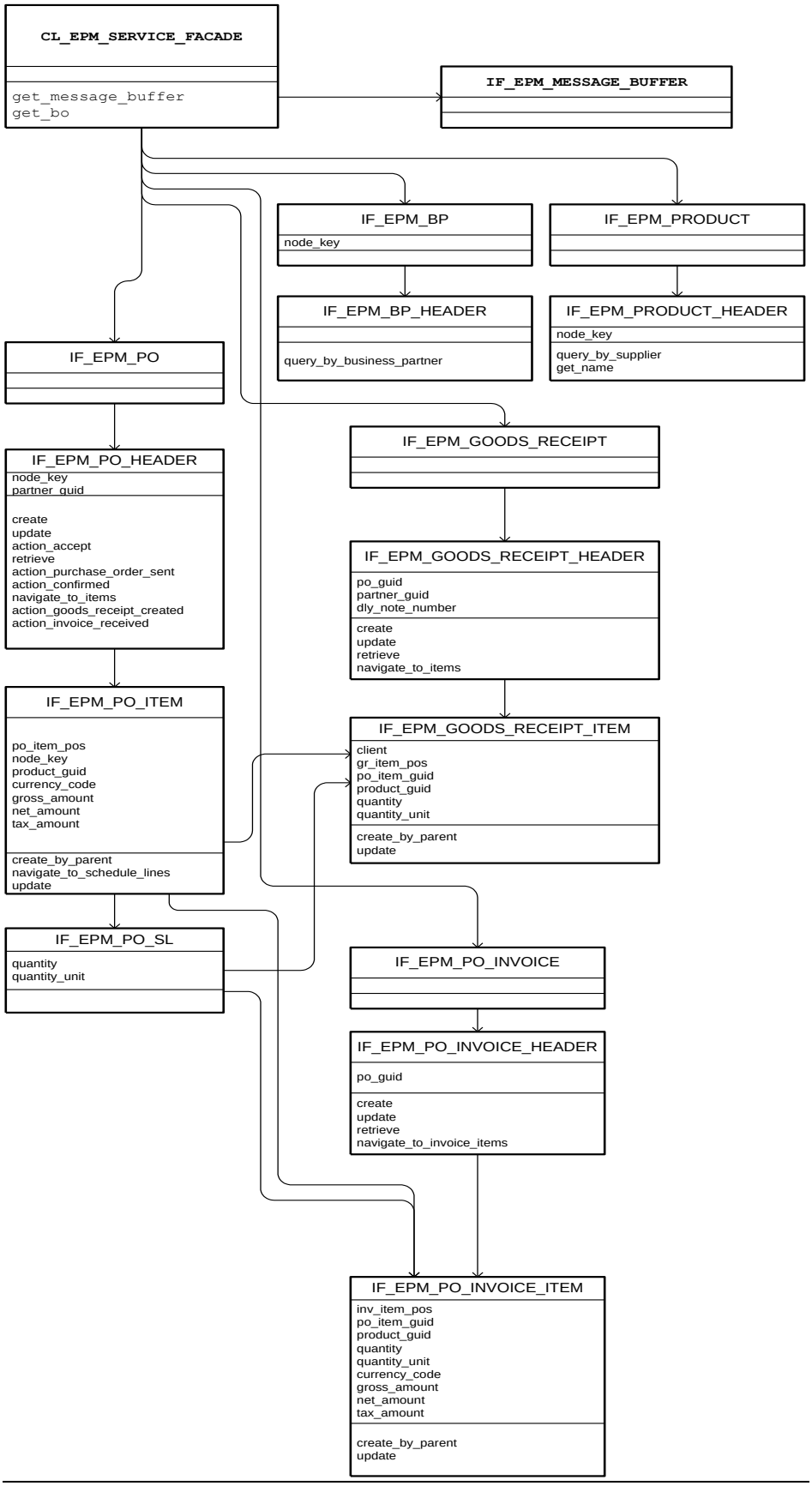
Hint: After invoking a method of a BO, the BO retrieve method must be used in order to get the latest status of the object.

Before running the sample code, use the EPM Data Generator to create master data. The tool can be accessed with the APAP transaction SEPM_DG and is described in The NetWeaver Enterprise Procurement Model - An Introduction.

The consistent source code is provided under “Source Code of the example”. The explanations of the implementation aspects do not show code.

The source code will be available as program RS_EPM_SAMPLE_PURCHASE in package S_EPM_SAMPLES starting with Release SAP NetWeaver 7.02 SP15, SAP NetWeaver 7.31 SP10, SAP NetWeaver 7.4 SP5.

Structure and navigation of the PO object and related objects:



EPM Service Facade

The EPM Service Facade is the pivot when working with EPM BOs. It performs the following tasks:

- Provides BO instances as singletons, such as a Purchase Order BO instance
- Provides a message buffer instance which is consumed by BO operations for reporting business problems to a BO consumer
- Triggers the transactional SAVE cycle which asks the EPM BOs to check their consistency. If no problems are detected, the BO data is persisted into the data base.

To work with an EPM Business Object, the consumer application requires an instance of the Business Object in question.

```
DATA: li_epm_po TYPE REF TO if_epm_po.
li_epm_po ?= cl_epm_service_facade=>get_bo( iv_bo_name =
    if_epm_po_header=>gc_bo_name ).
```

To save BO data, the service facade provides a method `save`. This checks the consistency of each BO in the ABAP session and checks whether it is ready to be persisted. If this is the case, the modified data of all BOs is stored in the corresponding DB tables by the BOs. If not, none of the modified BO data is persisted. Once the data has been stored by the BOs, the Service Facade performs a COMMIT WORK.

```
cl_epm_service_facade=>save(
  EXPORTING
    ii_message_buffer = li_message_buffer
  RECEIVING
    rv_success       = lv_success  ).
```

If successful, parameter rv_success will be set to ABAP_TRUE. If not, it will be set to ABAP_FALSE. If problems occur, error messages can be retrieved from the message buffer.

EPM Message Handling

An EPM Message Buffer is used to communicate between EPM BOs and consumers. A message is created by a BO which indicates that the BO has detected an error or has encountered a situation which the BO implementer thinks the consumer should be informed about. An example might be if data is locked by a different user. The messages are mainly set up in a way that a human user can understand. A consumer requests a message buffer instance from the EPM Service Facade and passes it over to a BO operation.

Get a message buffer instance

```
DATA: li_message_buffer     TYPE REF TO if_epm_message_buffer.
li_message_buffer = cl_epm_service_facade=>get_message_buffer( ).
```

Evaluate the messages in the buffer

```
DATA: lt_messages   TYPE if_epm_message_buffer=>tt_messages.
DATA: ls_message    TYPE if_epm_message_buffer=>ty_message.
DATA: lv_txterror   TYPE string.
lt_messages = li_message_buffer->get_messages( ).
LOOP AT lt_messages INTO ls_message.
  lv_txterror = ls_message->get_longtext( ).
  WRITE: / 'message text : ', lv_txterror.
ENDLOOP.
```

IMPLEMENTATION

Create a Purchase Order node element

Use the EPM Service Facade to create a typed singleton instance of the Purchase Order BO `if_epm_po`. The interface provides members and operations to work with the Purchase Order Header, Item and Schedule Line. The Header contains information such as the Business Partner (BP) and status information such as creator, creation date and order status. An item contains information such as product and price information. The Schedule Line contains information about the quantity and expected delivery period of a product.
Interface `if_epm_po_header` provides method `create`. This returns a Header object with default values. The mandatory fields PARTNER_GUID and the optional values NOTE_GUID and CURRENCY_CODE can be set using the `update` method. The first two values can be updated with special setter methods which will be shown in the sample. For method `create`, the sample shows how to evaluate the Message Buffer. To keep the sample code short, this is not done for the other node operations. It must be done in productive code however.

Set the Business Partner at the Purchase Order Header

A Purchase Order Header has field “PARTNER_GUID” as a mandatory property. This property identifies the Business Partner BO header node element of the supplier. We use method `set_business_partner_id` to provide the ID of the supplier. This method determines the PARTNER_GUID (node key) for this ID. See chapter 0 for more information about valid suppliers.
A Purchase Order only contains Items that can be supplied by one business partner. Each product is supplied by exactly one business partner. Checks are implemented to ensure that the PO is consistent in this regard. If a PO Header does not have a Business Partner set, it will be set to the supplier of the product that is defined for the first Item.

Set a note text at the Purchase Order Header

A Purchase Order Header can have a language-specific note text to provide additional information. The note text can be set using method `set_note`. This method creates a text BO header node element and a corresponding subnode element which carries the actual text. If the text is empty, the existing note is deleted. If the language is the original language, the notes are deleted for all languages. The key of the note text BO is available in the PO Header field NOTE_GUID once the text has been set.

Create a Purchase Order Item and Schedule Line

Interface `if_epm_po_item` provides method `create_by_parent`. This creates exactly one Item, filled with default values, and implicitly also creates exactly one Schedule Line for the Item. It also creates the value PO_ITEM_POS, which identifies an Item line at the parent PO. The values are counted in multiples of 10. The method returns the PO Item object.

Set update enabled values at the Item and Schedule Line

An Item has the mandatory field PRODUCT_GUID and the optional field NOTE_GUID. Method set_product_id converts the given Product ID into a node key and updates the PO Item field PRODUCT_GUID with this node key. This field is a mandatory property. A Purchase Order can only contain Items that can be supplied by exactly one business partner. Checks are made to ensure that the PO is consistent in this regard.

The implicitly created Schedule Line has the optional fields QUANTITY and DELIVERY_DATE, which can be set with method `update` of interface if_epm_po_sl. To do this, we first need the node element data of the Schedule Line first. The QUANTITY_UNIT field is filled with the corresponding value from the product object.

Purchase Order Status
The Purchase Order Header has five status fields. The field values are either provided with default values or are set according to “action” methods. For more information about the status types, see Purchase Order Statuses in the appendix.

Status Field Name	Status Values
LIFECYCLE_STATUS	New, In Process, Rejected, Cancelled, Closed
APPROVAL_STATUS	Initial, Approved, Rejected
CONFIRM_STATUS	Initial, Sent, Confirmed, Cancelled
ORDERING_STATUS	Initial, Delivered
INVOICING_STATUS	Initial, Invoiced

When it is created, a Purchase Order has the lifecycle status “New”. “New” indicates that the PO has not undergone any other processing. A member of the buyer team can accept or reject the Purchase Order. This changes the status to “In Process” or “Rejected”. Once the purchase order has been accepted, the approval status is set from “Initial” to “Approved”. Serving as an example, the accept action is executed on the Purchase Order in the sample code.

A Purchase Order has confirmation status “Initial” until it is sent to the supplier. Method `action_purchase_order_sent` changes the status to “Sent”. The status changes to “Confirmed” when the supplier confirms that the order can be executed. This is done with method `action_confirmed`. The status is set to canceled if the supplier refuses to execute the order request.

Goods Receipt and Items
Creation of a Goods Receipt instance is triggered by a Goods Receipt message. This indicates that ordered goods have been received by the company which ordered the products. Goods are stored directly in the local stock. Use the EPM service facade to create a typed singleton instance of the Goods Receipt BO `if_epm_goods_receipt`. The interface provides operations to work with the Goods Receipt Header and Item. Interface `if_epm_goods_receipt_header` provides method `create`, which returns a header object with default values. The mandatory fields PO_GUID, PARTNER_GUID and DLY_NOTE_NUMBER can be set using the method `update`. To update a Goods Receipt BO with the Purchase Order, the confirmation status for this PO must be “C*onfirmed”*.

Goods Receipt Item
Interface `if_epm_goods_receipt_item` provides method `create_by_parent`, which creates and returns an Item object filled with default values. The Item has the mandatory properties PO_ITEM_GUID, GR_ITEM_POS, QUANTITY, QUANTITY_UNIT, PRODUCT_GUID. Values for the fields are set with method `update`.

Purchase Order Invoice and Items
The Purchase Order Invoice is used to complete the purchasing process. An Invoice contains a Header (information about creator, partner and amounts) and Items (information about Item position, product and quantity). To create an Invoice, the Purchase Order must have lifecycle status “In Process”, confirmation status “Confirmed” and invoicing status “Initial”. Once a PO Invoice has been saved, the PO Invoice Header is read-only, and the PO Invoice Item operations `Create`, `Delete` and `Update` are disabled. When saving a PO Invoice (which can be done exactly once as in the previous sentence), action invoice_received is triggered at the Purchase Order. If action invoice_received has been triggered successfully, action `close` is triggered at the Purchase Order. This is only successful if the corresponding Goods Receipt has been created. If this is not the case, the action is not executed.

Use the EPM Service Facade to create a typed singleton instance of the Purchase Order Invoice BO if_epm_po_invoice. The interface provides operations to work with the Purchase Order Invoice. Use method `create` at interface if_epm_po_invoice_header to create an Invoice Header with default values. This method returns the created PO Invoice Header object. Then use method `update` to update the Invoice Header with the mandatory field PO_GUID, which is the Purchase Order node key.

Purchase Order Invoice Item
Method `create_by_parent` of interface `if_epm_po_invoice_item` creates an Invoice Item with default values. It returns the created PO Invoice Item object. The Item has the mandatory properties PO_ITEM_GUID, INV_ITEM_POS. Values for these fields and others are set with method `update`.

Save created data in database
Firstly, trigger the checking of all BOs that were created by the service façade. This check is performed with method `check` from the service facade. If all BO related checks are successful, it returns True. Otherwise it returns false. The Message Buffer object provides information about problems detected by the BOs. The Save operation is performed with method `save` from the Service Facade.

Retrieve Purchase Order, Goods Receipt, PO Invoice data
We now retrieve the data for the Purchase Order we have just created. We assume that we know the ID of the Purchase Order.

Purchase Order Header, Item and Schedule Line
To retrieve the PO Header data, we use method `retrieve` at the PO Header interface `if_epm_po_header`. This imports a table with Purchase Order node_keys. To obtain this table, you first need the node key for the Purchase Order ID you want to work with using method `convert_po_ids_to_keys`. Method `retrieve` exports a table of PO Header objects. These objects contain information like PO ID, gross/net/tax amount, status, node keys for the note and Business Partner. To get the Item of the Purchase Order we use method `navigate_to_items` at interface `if_epm_po_item`. This imports a table of PO node keys and exports a table of PO Item objects. These objects contain information such as the PO Item position, gross/net/tax amount, node keys for the note and product. The product node key is used to get the product information using method `navigate_to_product` at interface `if_epm_po_item`. To obtain the Schedule Line of the Item, use method `navigate_to_schedule_lines` which exports a list of PO Schedule Line objects. These objects contain information such as quantity and delivery-data.

Goods Receipt Header and Item
To get the Goods Receipt Header, we use method `navigate_to_goods_receipt` from the Purchase Order Header interface `if_epm_po_header`. This imports a table of PO node-keys (same as for navigation to PO Items) and exports a table of Goods Receipt Header objects. These objects contain information such as the PO node-key and Business Partner GUID. To get the Item of the Goods Receipt, we use method `navigate_to_items` of interface if_epm_goods_receipt_header. This imports a table of Goods Receipt node-keys and exports a table of Goods Receipt Item objects. These objects contain information such as PO Item GUID, Goods Receipt Item position and quantity.

Purchase Order Invoice Header and Item
To get the Invoice Header, we use the Purchase Order Header interface `if_epm_po_header` method `navigate_to_invoice`. This imports a table of PO node-keys (same as for navigation to PO Items) and exports a table of PO Invoice Header objects. These objects contain information such as the PO GUID, business partner GUID and amounts. To get the Item of the Invoice, we use method `navigate_to_invoice_items` at interface if_epm_po_invoice_header. This imports a table of Invoice Header node-keys and exports a table of Invoice Item objects. These objects contain information such as PO Invoice Item GUID, Invoice Item position, quantity and amount.

Retrieve data using query methods
Queries are used to retrieve data from a BO. Queries can have input parameters which describe the selection criteria to be used for data retrieval. The result of a query is the set of BO data that matches the selection which the consumer of the query sets.

Retrieve all Purchase Orders by company name
Interface `if_epm_po_header` has a method called `query_by_supplier`. One of the method's input parameters is a table of select options for company names. Enter the company name "SAP" here. The method has an output parameter with a table of Purchase Order Header objects and the number of rows found.

Retrieve Purchase Orders by company name and with approval status "Closed"
Interface `if_epm_po_header` has a method called `query_by_header`. One of the method's input parameters is a table of select options for company names (as in the previous query sample). Enter the company name "SAP" here. Another input parameter is a table of select options for lifecycle status. Enter the lifecycle status IF_EPM_PO_HEADER=> GC_LIFECYCLE_STATUS_CLOSED here. The method has an output parameter with a table of Purchase Order Header objects and the number of rows found.

Retrieve Business Partners by company name
Interface `if_epm_bp_header` has a method called `query_by_business_partner`. One of the method's input parameters is a table of select options for company names. Enter the company name "SAP" here. The method has an output parameter with a table of Business Partner Header objects and the number of rows found.

Retrieve Products of a Business Partner
This sample retrieves the Product objects of the Business Partner used in the previous query ("Retrieve Business Partners by company name").
Interface `if_epm_product_header` has a method called `query_by_supplier`. One of the method's input parameters is a table of select options for Business Partner IDs. Generate this table using output parameter `et_data` from the previous query. The method has an output parameter with a table of Product Header objects and the number of rows found.

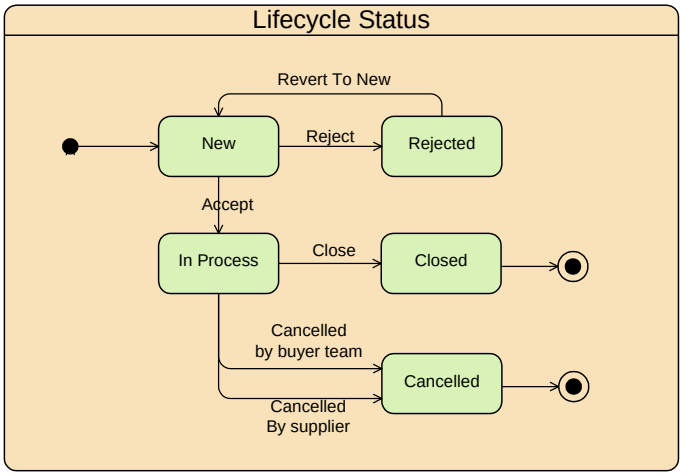
APPENDIX

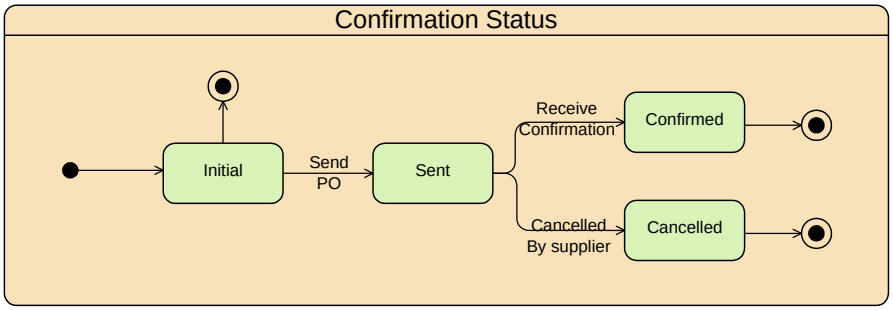
Predefined data for the sample code

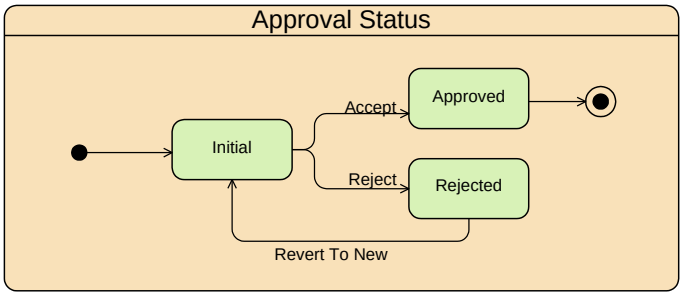
Supplier for Purchase Order:
Valid suppliers for a Purchase Order Header are the business partners in table SNWD_BPA. This tutorial uses the business partner "SAP" with the "BP_ID=0100000000". This field has the data type CHAR length 10. Note the leading 0s.

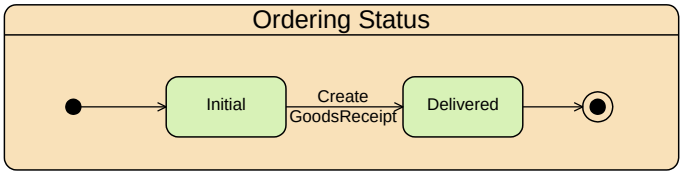
Product for Purchase Order Item:
Valid Products for a Purchase Order Item are the products in table SNWD_PD. This tutorial uses the product "Notebook Basic 15" with the "PRODUCT_ID=HT-1000". When a PRODUCT_ID is set to a Purchase Order Item, the system checks that the product is supplied from the business partner of the Purchase Order Header. If problems occur, check in table SNWD_PD that the product has the correct business partner "SAP". The relation is field NODE_KEY in table SNWD_BPA to field SUPPLIER_GUID in table SNWD_PD
A product has only one supplier (business partner).

Purchase Order Statuses









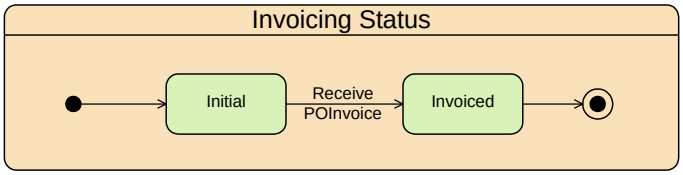
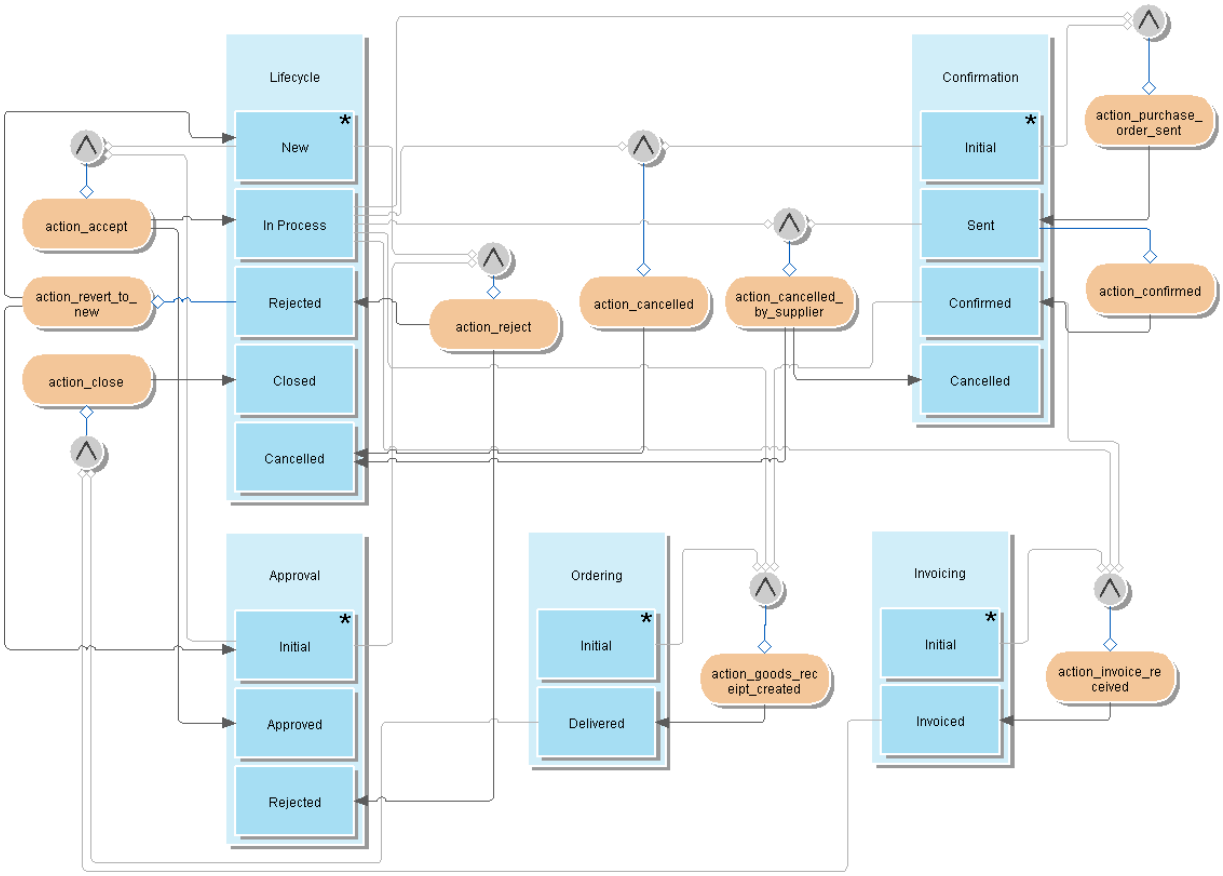


Diagram showing dependencies between status variables



Source Code of the example

This source code can be pasted into a standard ABAP program.

```
*&---------------------------------------------------------------------*
*& Report RS_EPM_SAMPLE_PURCHASE
*&
*&---------------------------------------------------------------------*
*& This report is a sample to show the handling of a
*& Purchase Order with Items and Schedule Lines,
*& Purchase Order Goods Receipt with Item,
*& Purchase Order Invoice with Item,
*& retrieve a Purchase Order and related nodes
*& query of Purchase Orders.
*&---------------------------------------------------------------------*

REPORT rs_epm_sample_purchase.
DATA lv_success        TYPE abap_bool.
DATA li_message_buffer TYPE REF TO if_epm_message_buffer.
DATA li_epm_po         TYPE REF TO if_epm_po.
DATA ls_po_header_data TYPE if_epm_po_header=>ty_node_data.
DATA lt_po_hd_keys     TYPE if_epm_bo=>tt_node_keys.
DATA lt_po_item_keys   TYPE if_epm_bo=>tt_node_keys.

WRITE: / 'Create new Purchase Order with nodes'. "#EC NOTEXT
* setup message buffer
li_message_buffer = cl_epm_service_facade=>get_message_buffer( ).

* get a singleton instance of the purchase order business object.
li_epm_po ?= cl_epm_service_facade=>get_bo( iv_bo_name = if_epm_po_header=>gc_bo_name ).

* PO Header create with defaults values
li_epm_po->if_epm_po_header~create(
  EXPORTING
    ii_message_buffer = li_message_buffer
  IMPORTING
    es_data           = ls_po_header_data
    ev_success        = lv_success ).
WRITE: / 'PO create success = ', lv_success. "#EC NOTEXT

* Evaluate the message buffer
DATA lt_messages TYPE if_epm_message_buffer=>tt_messages.
DATA ls_message  TYPE if_epm_message_buffer=>ty_message.
DATA lv_txterror TYPE string.

lt_messages = li_message_buffer->get_messages( ).
LOOP AT lt_messages INTO ls_message.
  lv_txterror = ls_message->get_longtext( ).
  WRITE: / 'PO create - message text : ', lv_txterror. "#EC NOTEXT
ENDLOOP.

* PO Header update
DATA lt_bp_data TYPE if_epm_bp_header=>tt_node_data.
DATA ls_bp_data TYPE if_epm_bp_header=>ty_node_data.

* Set the Business Partner at the Purchase Order Header
*     Use a given Business-Partner ID, which is SAP
li_epm_po->if_epm_po_header~set_business_partner_id(
  EXPORTING
    iv_node_key             = ls_po_header_data-node_key
    iv_business_partner_id  = '0100000000'
    ii_message_buffer       = li_message_buffer
  IMPORTING
    ev_success              = lv_success ).
WRITE: / 'PO Header set set_business_partner_id success = ', lv_success. "#EC NOTEXT

INSERT ls_po_header_data-node_key INTO TABLE lt_po_hd_keys.

li_epm_po->if_epm_po_header~navigate_to_business_partner(
  EXPORTING
    it_source_node_keys     = lt_po_hd_keys
    ii_message_buffer       = li_message_buffer
  IMPORTING
```

```
    et_data                 = lt_bp_data ).
READ TABLE lt_bp_data INDEX 1 INTO ls_bp_data.
WRITE: / 'PO Business Partner', ls_bp_data-bp_id, ls_bp_data-company_name, ls_bp_data-node_key. "#EC
NOTEXT

* Set a note text at the Purchase Order Header
li_epm_po->if_epm_po_header~set_note(
  EXPORTING
    iv_node_key             = ls_po_header_data-node_key
    iv_text                 = 'SAP Netweaver EPM - Tutorial for Purchase orders' "#EC NOTEXT
    ii_message_buffer       = li_message_buffer
  IMPORTING
    ev_success              = lv_success ).
WRITE: / 'PO Header set set_note success = ', lv_success. "#EC NOTEXT

* Create Purchase Order Item and Schedule Line
DATA ls_po_item_data   TYPE if_epm_po_item=>ty_node_data.
DATA lt_po_sl_data     TYPE if_epm_po_sl=>tt_node_data.
DATA lt_node_key_info  TYPE if_epm_bo=>tt_node_key_info.
DATA ls_node_key_info  TYPE if_epm_bo=>ty_node_key_info.
DATA lt_pd_header_data TYPE if_epm_product_header=>tt_node_data.
DATA ls_pd_header_data TYPE if_epm_product_header=>ty_node_data.
DATA lv_delivery_day   TYPE sydatum.
DATA lv_tz             TYPE ttzz-tzone.
FIELD-SYMBOLS <ls_po_sl_data> TYPE if_epm_po_sl=>ty_node_data.

li_epm_po->if_epm_po_item~create_by_parent(
  EXPORTING
    iv_source_node_key = ls_po_header_data-node_key
    ii_message_buffer  = li_message_buffer
  IMPORTING
    es_data            = ls_po_item_data
    ev_success         = lv_success ).
WRITE: / 'PO Item create success = ', lv_success. "#EC NOTEXT

* Set the product of the item
li_epm_po->if_epm_po_item~set_product_id(
  EXPORTING
    iv_node_key             = ls_po_item_data-node_key
    iv_product_id           = 'HT-1000'
    ii_message_buffer       = li_message_buffer
  IMPORTING
    ev_success              = lv_success ).
WRITE: / 'PO Item set_product_id success = ', lv_success. "#EC NOTEXT

INSERT ls_po_item_data-node_key INTO TABLE lt_po_item_keys.

* Get the information about the product object
li_epm_po->if_epm_po_item~navigate_to_product(
  EXPORTING
    it_source_node_keys     = lt_po_item_keys
    ii_message_buffer       = li_message_buffer
  IMPORTING
    et_data                 = lt_pd_header_data ).
READ TABLE lt_pd_header_data INDEX 1 INTO ls_pd_header_data.
WRITE: 'PO Item Product', ls_pd_header_data-product_id, ls_pd_header_data-node_key. "#EC NOTEXT

* Update schedule line of item
*   first get the schedule line nodes of the item
li_epm_po->if_epm_po_item~navigate_to_schedule_lines(
  EXPORTING
    it_source_node_keys = lt_po_item_keys
    ii_message_buffer   = li_message_buffer
  IMPORTING
    et_data             = lt_po_sl_data ).

READ TABLE lt_po_sl_data INDEX 1 ASSIGNING <ls_po_sl_data>.
<ls_po_sl_data>-quantity = 10.
lv_delivery_day          = sy-datlo + 10.
lv_tz = ''.
CONVERT DATE lv_delivery_day INTO TIME STAMP <ls_po_sl_data>-delivery_date TIME ZONE lv_tz.

li_epm_po->if_epm_po_sl~update(
  EXPORTING
```

```
    it_data           = lt_po_sl_data
    ii_message_buffer = li_message_buffer
  IMPORTING
    et_node_key_info  = lt_node_key_info ).

READ TABLE lt_node_key_info INDEX 1 INTO ls_node_key_info.
WRITE: / 'PO Schedule Line update success = ', ls_node_key_info-operation_success, ls_node_key_info-
error_info. "#EC NOTEXT

*---------------------------------------------------------------------*
* PO Status
* change lifecycle status to "In Process", this sets the approval status to approved as well
* change the confirmation status to "sent" and then to "confirmed"
li_epm_po->if_epm_po_header~action_accept(
  EXPORTING
    it_node_keys      = lt_po_hd_keys
    ii_message_buffer = li_message_buffer
  IMPORTING
    et_node_key_info  = lt_node_key_info ).

READ TABLE lt_node_key_info INDEX 1 INTO ls_node_key_info.
WRITE: / 'PO Header action_accept success = ', ls_node_key_info-operation_success, ls_node_key_info-
error_info. "#EC NOTEXT

* Set the Purchase Order confirmation status to sent.
li_epm_po->if_epm_po_header~action_purchase_order_sent(
  EXPORTING
    it_node_keys      = lt_po_hd_keys
    ii_message_buffer = li_message_buffer
  IMPORTING
    et_node_key_info  = lt_node_key_info ).

READ TABLE lt_node_key_info INDEX 1 INTO ls_node_key_info.
WRITE: / 'PO Header action_purchase_order_sent success = ', ls_node_key_info-operation_success,
ls_node_key_info-error_info. "#EC NOTEXT

* Set the Purchase order confirmation status to confirmed.
li_epm_po->if_epm_po_header~action_confirmed(
  EXPORTING
    it_node_keys      = lt_po_hd_keys
    ii_message_buffer = li_message_buffer
  IMPORTING
    et_node_key_info  = lt_node_key_info ).

READ TABLE lt_node_key_info INDEX 1 INTO ls_node_key_info.
WRITE: / 'PO Header action_confirmed success = ', ls_node_key_info-operation_success,
ls_node_key_info-error_info. "#EC NOTEXT

*&---------------------------------------------------------------------*
*&  Goods Receipt and Items
*&---------------------------------------------------------------------*
DATA li_epm_gr                   TYPE REF TO if_epm_goods_receipt.
DATA ls_goodsreceipt_header_data TYPE if_epm_goods_receipt_header=>ty_node_data.
DATA lt_goodsreceipt_header_data TYPE if_epm_goods_receipt_header=>tt_node_data.
DATA lt_goodsreceipt_item_data   TYPE if_epm_goods_receipt_item=>tt_node_data.
DATA ls_goodsreceipt_item_data   TYPE if_epm_goods_receipt_item=>ty_node_data.

* get goods receipt singleton
li_epm_gr ?= cl_epm_service_facade=>get_bo( iv_bo_name = if_epm_goods_receipt=>gc_bo_name ).

li_epm_gr->if_epm_goods_receipt_header~create(
  EXPORTING
    ii_message_buffer = li_message_buffer
  IMPORTING
    es_data           = ls_goodsreceipt_header_data
    ev_success        = lv_success ).
WRITE:/ 'Goods Receipt Header Create, success = ', lv_success. "#EC NOTEXT

* GR update Header-Data
ls_goodsreceipt_header_data-po_guid = ls_po_header_data-node_key.
ls_goodsreceipt_header_data-partner_guid = ls_bp_data-node_key.
ls_goodsreceipt_header_data-dly_note_number = '0123456789'.
INSERT ls_goodsreceipt_header_data INTO TABLE lt_goodsreceipt_header_data.
```

```
li_epm_gr->if_epm_goods_receipt_header~update(
  EXPORTING
    it_data           = lt_goodsreceipt_header_data
    ii_message_buffer = li_message_buffer
  IMPORTING
    et_node_key_info  = lt_node_key_info ).

READ TABLE lt_node_key_info INDEX 1 INTO ls_node_key_info.
WRITE: / 'Goods Receipt update success = ', ls_node_key_info-operation_success, ls_node_key_info-
error_info. "#EC NOTEXT

li_epm_gr->if_epm_goods_receipt_item~create_by_parent(
  EXPORTING
    iv_source_node_key = ls_goodsreceipt_header_data-node_key
    ii_message_buffer  = li_message_buffer
  IMPORTING
    es_data            = ls_goodsreceipt_item_data
    ev_success         = lv_success ).
WRITE:/ 'Goods Receipt Item create_by_parent success = ', lv_success. "#EC NOTEXT

* For simplification the data for goods receipt item are taken from PO-item
ls_goodsreceipt_item_data-gr_item_pos = ls_po_item_data-po_item_pos.
ls_goodsreceipt_item_data-po_item_guid = ls_po_item_data-node_key.
ls_goodsreceipt_item_data-product_guid = ls_pd_header_data-node_key.
ls_goodsreceipt_item_data-quantity = <ls_po_sl_data>-quantity.
ls_goodsreceipt_item_data-quantity_unit = <ls_po_sl_data>-quantity_unit.
INSERT ls_goodsreceipt_item_data INTO TABLE lt_goodsreceipt_item_data.

li_epm_gr->if_epm_goods_receipt_item~update(
  EXPORTING
    it_data           = lt_goodsreceipt_item_data
    ii_message_buffer = li_message_buffer
  IMPORTING
    et_node_key_info  = lt_node_key_info ).

READ TABLE lt_node_key_info INDEX 1 INTO ls_node_key_info.
WRITE: / 'Goods Receipt Item update success = ', ls_node_key_info-operation_success,
ls_node_key_info-error_info. "#EC NOTEXT

*&---------------------------------------------------------------------*
*&  Purchase Order Invoice and Items
*&---------------------------------------------------------------------*
DATA li_epm_invoice       TYPE REF TO if_epm_po_invoice.
DATA lt_invoice_node_data TYPE if_epm_po_invoice_header=>tt_node_data.
DATA ls_invoice_node_data TYPE if_epm_po_invoice_header=>ty_node_data.
DATA ls_invoice_item_data TYPE if_epm_po_invoice_item=>ty_node_data.
DATA lt_invoice_item_data TYPE if_epm_po_invoice_item=>tt_node_data.
DATA lt_po_item_data      TYPE if_epm_po_item=>tt_node_data.
DATA lt_po_hd_data        TYPE if_epm_po_header=>tt_node_data.

* get PO invoice singleton
li_epm_invoice ?= cl_epm_service_facade=>get_bo( iv_bo_name = if_epm_po_invoice=>gc_bo_name ).

li_epm_invoice->if_epm_po_invoice_header~create(
  EXPORTING
    ii_message_buffer = li_message_buffer
  IMPORTING
    es_data           = ls_invoice_node_data
    ev_success        = lv_success ).
WRITE: / 'PO Invoice header create success = ', lv_success. "#EC NOTEXT

*&---------------------------------------------------------------------*
*& Set invoice header data
*& For simplicity the values for the invoice are taken from the purchase order not from real source.
*& If the po invoice header values don't correspond to the po header values then error information
*& is provided in the object ii_message_buffer.
*& Get up-to-date data of the PO header first
li_epm_po->if_epm_po_header~retrieve(
  EXPORTING
    it_node_keys      = lt_po_hd_keys
    ii_message_buffer = li_message_buffer
  IMPORTING
    et_data           = lt_po_hd_data ).
```

```
READ TABLE lt_po_hd_data INDEX 1 INTO ls_po_header_data.
ls_invoice_node_data-po_guid = ls_po_header_data-node_key.
ls_invoice_node_data-partner_guid = ls_po_header_data-partner_guid.
ls_invoice_node_data-currency_code = ls_po_header_data-currency_code.
ls_invoice_node_data-gross_amount = ls_po_header_data-gross_amount.
ls_invoice_node_data-net_amount = ls_po_header_data-net_amount.
ls_invoice_node_data-tax_amount = ls_po_header_data-tax_amount.

INSERT ls_invoice_node_data INTO TABLE lt_invoice_node_data.

li_epm_invoice->if_epm_po_invoice_header~update(
  EXPORTING
    it_data            =  lt_invoice_node_data
    ii_message_buffer  =  li_message_buffer
  IMPORTING
    et_node_key_info   =  lt_node_key_info ).

READ TABLE lt_node_key_info INDEX 1 INTO ls_node_key_info.
WRITE: / 'PO Invoice header update success = ', ls_node_key_info-operation_success,
ls_node_key_info-error_info. "#EC NOTEXT

* Create Invoice Item
li_epm_invoice->if_epm_po_invoice_item~create_by_parent(
  EXPORTING
    iv_source_node_key = ls_invoice_node_data-node_key
    ii_message_buffer  = li_message_buffer
  IMPORTING
    es_data            = ls_invoice_item_data
    ev_success         = lv_success ).
WRITE: / 'PO Invoice item create_by_parent success = ', lv_success. "#EC NOTEXT

*&---------------------------------------------------------------------*
*& For simplicity the values for the invoice are taken from the purchase order not from real source.
*& If the po invoice header values don't correspond to the po header values then error information
*& is provided in the object ii_message_buffer.
*& Get up-to-date data of the po item first.
li_epm_po->if_epm_po_item~retrieve(
  EXPORTING
    it_node_keys      = lt_po_item_keys
    ii_message_buffer = li_message_buffer
  IMPORTING
    et_data           = lt_po_item_data ).
READ TABLE lt_po_item_data INDEX 1 INTO ls_po_item_data.

* Set PO-item data to Invoice-item data
ls_invoice_item_data-inv_item_pos = ls_po_item_data-po_item_pos.
ls_invoice_item_data-po_item_guid = ls_po_item_data-node_key.
ls_invoice_item_data-product_guid = ls_pd_header_data-node_key.
ls_invoice_item_data-quantity = <ls_po_sl_data>-quantity.
ls_invoice_item_data-quantity_unit = <ls_po_sl_data>-quantity_unit.
ls_invoice_item_data-currency_code = ls_po_item_data-currency_code.
ls_invoice_item_data-gross_amount = ls_po_item_data-gross_amount.
ls_invoice_item_data-net_amount = ls_po_item_data-net_amount.
ls_invoice_item_data-tax_amount = ls_po_item_data-tax_amount.
INSERT ls_invoice_item_data INTO TABLE lt_invoice_item_data.

*  Update Invoice-Item
li_epm_invoice->if_epm_po_invoice_item~update(
  EXPORTING
    it_data            = lt_invoice_item_data
    ii_message_buffer  = li_message_buffer
  IMPORTING
    et_node_key_info   = lt_node_key_info ).

READ TABLE lt_node_key_info INDEX 1 INTO ls_node_key_info.
WRITE: / 'PO Invoice Item update success = ', ls_node_key_info-operation_success, ls_node_key_info-
error_info. "#EC NOTEXT

*&---------------------------------------------------------------------*
*& Save created data in database
*&---------------------------------------------------------------------*

*       Trigger the checking of the contexts  *
*       of all BOs that have been created by  *
```

```
*       the Service Facade
lv_success = cl_epm_service_facade=>check( li_message_buffer ).
WRITE: / 'PO and releated data checked = ', lv_success. "#EC NOTEXT

lv_success = cl_epm_service_facade=>save( iv_cleanup = abap_false ii_message_buffer =
li_message_buffer ).
WRITE: / 'PO and releated data saved = ', lv_success. "#EC NOTEXT
ULINE.

*&---------------------------------------------------------------------*
*& Retrieve Purchase Order, Goods Receipt, PO Invoice data
*&---------------------------------------------------------------------*
DATA lt_po_ids         TYPE if_epm_po_header=>tt_po_identifiers.
DATA lt_po_key_mapping TYPE if_epm_bo=>tt_key_mapping_info.
DATA ls_po_key_mapping TYPE if_epm_bo=>ty_key_mapping_info.
DATA lt_po_header_data TYPE if_epm_po_header=>tt_node_data.
DATA ls_bo_text        TYPE snwd_desc.
DATA ls_bp_header_data TYPE if_epm_bp_header=>ty_node_data.
DATA lt_bp_header_data TYPE if_epm_bp_header=>tt_node_data.
DATA lt_product_data   TYPE if_epm_product_header=>tt_node_data.
DATA ls_product_data   TYPE if_epm_product_header=>ty_node_data.

CLEAR lt_po_ids.
CLEAR lt_po_item_keys.

WRITE / 'Purchase Order retrieve'. "#EC NOTEXT
* Retrieve the data of the previously created purchase order
APPEND ls_po_header_data-po_id TO lt_po_ids.

* Get the node_key for the available ID.
li_epm_po->if_epm_po_header~convert_po_ids_to_keys(
  EXPORTING
    it_key_values   = lt_po_ids
   IMPORTING
     et_key_mapping = lt_po_key_mapping ).

READ TABLE lt_po_key_mapping INDEX 1 INTO ls_po_key_mapping.

CLEAR  lt_po_hd_keys.
APPEND ls_po_key_mapping-node_key TO lt_po_hd_keys.

* Retrieve EPM PO header data
li_epm_po->if_epm_po_header~retrieve(
  EXPORTING
    it_node_keys            = lt_po_hd_keys
    iv_edit_mode            = if_epm_bo=>gc_read_only
    ii_message_buffer       = li_message_buffer
  IMPORTING
    et_data                 = lt_po_header_data ).

READ TABLE lt_po_header_data INDEX 1 INTO ls_po_header_data.
WRITE: / 'PO Header: ', ls_po_header_data-node_key, ls_po_header_data-po_id, ls_po_header_data-
node_key, "#EC NOTEXT
                       ls_po_header_data-gross_amount CURRENCY ls_po_header_data-currency_code,
ls_po_header_data-currency_code. "#EC NOTEXT
WRITE: /5 'lifecycle_status', 35 ls_po_header_data-lifecycle_status.  "#EC NOTEXT
WRITE: /5 'approval_status', 35 ls_po_header_data-approval_status.    "#EC NOTEXT
WRITE: /5 'confirm_status', 35 ls_po_header_data-confirm_status.      "#EC NOTEXT
WRITE: /5 'ordering_status', 35 ls_po_header_data-ordering_status.    "#EC NOTEXT
WRITE: /5 'invoicing_status', 35 ls_po_header_data-invoicing_status.  "#EC NOTEXT

* Retrieve PO header note text
ls_bo_text = li_epm_po->if_epm_po_header~get_note( iv_node_key  = ls_po_header_data-node_key
ii_message_buffer = li_message_buffer ). "#EC NOTEXT
WRITE:  / 'PO Header Note: ', ls_bo_text. "#EC NOTEXT

* Retrieve supplier (business partner) of the PO header
li_epm_po->if_epm_po_header~navigate_to_business_partner(
  EXPORTING
    it_source_node_keys = lt_po_hd_keys
    ii_message_buffer   = li_message_buffer
  IMPORTING
    et_data             = lt_bp_header_data ).
```

```
READ TABLE lt_bp_header_data INDEX 1 INTO ls_bp_header_data.
WRITE: / 'PO Supplier: ', ls_bp_header_data-node_key, ls_bp_header_data-company_name. "#EC NOTEXT

* Get PO-ITEMS
li_epm_po->if_epm_po_header~navigate_to_items(
  EXPORTING
    it_source_node_keys     = lt_po_hd_keys
    ii_message_buffer       = li_message_buffer
  IMPORTING
    et_data                 = lt_po_item_data ).

READ TABLE lt_po_item_data INDEX 1 INTO ls_po_item_data.
WRITE: / 'PO Item: ', ls_po_item_data-node_key, ls_po_item_data-po_item_pos, "#EC NOTEXT
                      ls_po_item_data-gross_amount CURRENCY ls_po_item_data-currency_code,
ls_po_item_data-currency_code,
                      ls_po_item_data-net_amount CURRENCY ls_po_item_data-currency_code,
ls_po_item_data-currency_code,
                      ls_po_item_data-tax_amount CURRENCY ls_po_item_data-currency_code,
ls_po_item_data-currency_code,
                      ls_po_item_data-product_guid.

APPEND ls_po_item_data-node_key TO lt_po_item_keys.

* Get product of PO-Item
li_epm_po->if_epm_po_item~navigate_to_product(
  EXPORTING
    it_source_node_keys     = lt_po_item_keys
    ii_message_buffer       = li_message_buffer
  IMPORTING
    et_data                 = lt_product_data ).

READ TABLE lt_product_data INDEX 1 INTO ls_product_data.
WRITE: / 'PO Item Product: ', ls_product_data-node_key, ls_product_data-product_id, "#EC NOTEXT
                              ls_product_data-price CURRENCY ls_product_data-currency_code,
ls_product_data-currency_code,
                              ls_product_data-product_pic_url .

* Get schedule line of PO-Item
li_epm_po->if_epm_po_item~navigate_to_schedule_lines(
  EXPORTING
    it_source_node_keys     = lt_po_item_keys
    ii_message_buffer       = li_message_buffer
  IMPORTING
    et_data                 = lt_po_sl_data ).

READ TABLE lt_po_sl_data INDEX 1 ASSIGNING <ls_po_sl_data>.
WRITE: / 'PO Schedule Line: ', <ls_po_sl_data>-node_key, <ls_po_sl_data>-delivery_date,
<ls_po_sl_data>-quantity, <ls_po_sl_data>-quantity_unit. "#EC NOTEXT

* Get Goods receipt header
DATA lt_gr_header_data TYPE if_epm_goods_receipt_header=>tt_node_data.
DATA ls_gr_header_data TYPE if_epm_goods_receipt_header=>ty_node_data.
DATA lt_gr_keys        TYPE if_epm_bo=>tt_node_keys.
DATA lt_gr_item_data   TYPE if_epm_goods_receipt_item=>tt_node_data.
DATA ls_gr_item_data   TYPE if_epm_goods_receipt_item=>ty_node_data.

li_epm_po->if_epm_po_header~navigate_to_goods_receipt(
  EXPORTING
    it_source_node_keys     = lt_po_hd_keys
    ii_message_buffer       = li_message_buffer
  IMPORTING
    et_data                 = lt_gr_header_data ).

READ TABLE lt_gr_header_data INDEX 1 INTO ls_gr_header_data.
WRITE: / 'Goods Receipt Header: ', ls_gr_header_data-node_key, ls_gr_header_data-created_at,
ls_gr_header_data-po_guid. "#EC NOTEXT

* Get Goods receipt Item
INSERT ls_gr_header_data-node_key INTO TABLE lt_gr_keys.

li_epm_gr->if_epm_goods_receipt_header~navigate_to_items(
  EXPORTING
    it_source_node_keys     = lt_gr_keys
    ii_message_buffer       = li_message_buffer
```

```
  IMPORTING
    et_data                 = lt_gr_item_data ).
READ TABLE lt_gr_item_data INDEX 1 INTO ls_gr_item_data.
WRITE: / 'Goods Receipt Item: ', ls_gr_item_data-node_key, ls_gr_item_data-gr_item_pos,
ls_gr_item_data-quantity, ls_gr_item_data-quantity_unit. "#EC NOTEXT

* Get Purchase Order Invoice Header
DATA lt_po_invoice_header_data TYPE if_epm_po_invoice_header=>tt_node_data.
DATA ls_po_invoice_header_data TYPE if_epm_po_invoice_header=>ty_node_data.
DATA lt_po_invoice_header_keys TYPE if_epm_bo=>tt_node_keys.
DATA lt_po_invoice_item_data   TYPE if_epm_po_invoice_item=>tt_node_data.
DATA ls_po_invoice_item_data   TYPE if_epm_po_invoice_item=>ty_node_data.

li_epm_po->if_epm_po_header~navigate_to_invoice(
  EXPORTING
    it_source_node_keys     = lt_po_hd_keys
    ii_message_buffer       = li_message_buffer
  IMPORTING
    et_data                 = lt_po_invoice_header_data ).

READ TABLE lt_po_invoice_header_data INDEX 1 INTO ls_po_invoice_header_data.
WRITE: / 'PO Invoice Header: ', ls_po_invoice_header_data-node_key, "#EC NOTEXT
                                ls_po_invoice_header_data-gross_amount CURRENCY
ls_po_invoice_header_data-currency_code, ls_po_invoice_header_data-currency_code,
                                ls_po_invoice_header_data-net_amount CURRENCY
ls_po_invoice_header_data-currency_code, ls_po_invoice_header_data-currency_code,
                                ls_po_invoice_header_data-tax_amount CURRENCY
ls_po_invoice_header_data-currency_code, ls_po_invoice_header_data-currency_code.
INSERT ls_po_invoice_header_data-node_key INTO TABLE lt_po_invoice_header_keys.

li_epm_invoice->if_epm_po_invoice_header~navigate_to_invoice_items(
  EXPORTING
    it_source_node_keys     = lt_po_invoice_header_keys
    ii_message_buffer       = li_message_buffer
  IMPORTING
    et_data                 = lt_po_invoice_item_data ).

READ TABLE lt_po_invoice_item_data INDEX 1 INTO ls_po_invoice_item_data.
WRITE: / 'PO Invoice Item: ', ls_po_invoice_item_data-node_key, "#EC NOTEXT
                              ls_po_invoice_item_data-gross_amount CURRENCY ls_po_invoice_item_data-
currency_code, ls_po_invoice_item_data-currency_code,
                              ls_po_invoice_item_data-net_amount CURRENCY ls_po_invoice_item_data-
currency_code, ls_po_invoice_item_data-currency_code,
                              ls_po_invoice_item_data-tax_amount CURRENCY ls_po_invoice_item_data-
currency_code, ls_po_invoice_item_data-currency_code.

ULINE.
*&---------------------------------------------------------------------*
*& Retrieve Data using Query methods
*&---------------------------------------------------------------------*

* Retrieve all Purchase Orders by company-name.
DATA lt_par_company_names TYPE if_epm_po_header=>tt_sel_par_company_names.
DATA ls_par_company_name  LIKE LINE OF lt_par_company_names.
DATA lv_nr_of_rows_found  TYPE if_epm_bo=>ty_query_nr_of_rows_found.

ls_par_company_name-low    = 'SAP'.
ls_par_company_name-sign   = 'I'.
ls_par_company_name-option = 'EQ'.
INSERT ls_par_company_name INTO TABLE lt_par_company_names.

li_epm_po->if_epm_po_header~query_by_supplier(
  EXPORTING
    it_sel_par_company_names = lt_par_company_names
  IMPORTING
    et_data                  = lt_po_header_data
    ev_nr_of_rows_found      = lv_nr_of_rows_found  ).
WRITE: / 'Query PO by company name "SAP", count = ', lv_nr_of_rows_found. "#EC NOTEXT

* Retrieve Purchase Orders gy company-name and with approval-status "Closed"
DATA lt_par_po_lifecycle_status TYPE if_epm_po_header=>tt_sel_par_lifecycle_status.
DATA ls_par_po_lifecycle_status LIKE LINE OF lt_par_po_lifecycle_status.

*ls_par_po_lifecycle_status-low    = 'C'.
```

```
ls_par_po_lifecycle_status-low    = IF_EPM_PO_HEADER=>GC_LIFECYCLE_STATUS_CLOSED.
ls_par_po_lifecycle_status-sign   = 'I'.
ls_par_po_lifecycle_status-option = 'EQ'.
INSERT ls_par_po_lifecycle_status INTO TABLE lt_par_po_lifecycle_status.

li_epm_po->if_epm_po_header~query_by_header(
  EXPORTING
    it_sel_par_company_names       = lt_par_company_names
    it_sel_par_lifecycle_status    = lt_par_po_lifecycle_status
  IMPORTING
    et_data                        = lt_po_header_data
    ev_nr_of_rows_found            = lv_nr_of_rows_found ).
WRITE: / 'Query PO by lifecycle status "Closed", count = ', lv_nr_of_rows_found. "#EC NOTEXT

* Retrieve Business Partners by company-name
DATA li_epm_bp TYPE REF TO if_epm_bp.

li_epm_bp ?= cl_epm_service_facade=>get_bo( iv_bo_name = if_epm_bp_header=>gc_bo_name ).

li_epm_bp->if_epm_bp_header~query_by_business_partner(
  EXPORTING
    it_sel_par_company_names       = lt_par_company_names
  IMPORTING
    et_data                        = lt_bp_data
    ev_nr_of_rows_found            = lv_nr_of_rows_found ).
WRITE: / 'Query BP by company name "SAP", count = ', lv_nr_of_rows_found. "#EC NOTEXT

* Retrieve Products of a Business Partner
DATA li_epm_product      TYPE REF TO if_epm_product.
DATA lt_par_bp_id        TYPE if_epm_product_header=>tt_sel_par_partner_ids.
DATA ls_par_bp_id        LIKE LINE OF lt_par_bp_id.
DATA lt_prod_header_data TYPE if_epm_product_header=>tt_node_data  ##NEEDED.

li_epm_product ?= cl_epm_service_facade=>get_bo( iv_bo_name = if_epm_product_header=>gc_bo_name ).

LOOP AT lt_bp_data INTO ls_bp_data.
  ls_par_bp_id-low    = ls_bp_data-bp_id.
  ls_par_bp_id-sign   = 'I'.
  ls_par_bp_id-option = 'EQ'.
  APPEND ls_par_bp_id TO lt_par_bp_id.
ENDLOOP.

li_epm_product->if_epm_product_header~query_by_supplier(
  EXPORTING
    it_sel_par_partner_ids = lt_par_bp_id
  IMPORTING
    et_data                = lt_prod_header_data
    ev_nr_of_rows_found    = lv_nr_of_rows_found ).
WRITE: / 'Query Products by Business Partner "SAP", count = ', lv_nr_of_rows_found. "#EC NOTEXT
```

www.sap.com