Datastore functions and procedures

Status: 2.24.3

- Functions and procedures starting with `f_geko_` and `f_tako_` are interface functions for the GEKO and TAKO robots do not belong to the BPS core.
- Functions and procedures starting with `cf_` or `cp_` respectively, are custom functions and procedures and do not belong to the BPS core. Ask your system integrator for documentation of such objects in case.

Function Reference

f_allpackjobusers

```
f_allpackjobusers(a_key in number, a_sep in varchar)
return varchar
```

```
Get list of all users holding locks on a packing job.
The separator text (for example a newline char or a comma) is given in
c_sep.
```

f_artattributename

```
f_artattributename(a_key number)
return varchar
```

Get the name (column c_name) of an article class. This is a convenience function to avoid joins in relation table routines.

f_articlefieldid

```
f_articlefieldid(a_key number)
return varchar
```

Get the id (column c_id) of an article field. This is a convenience function to avoid joins in relation table routines.

f_articlefielddidname

```
f_articlefielddidname(a_key number)
return varchar
```

Get the id and name (c_id||' '||c_name) of an article field. This is a convenience function to avoid joins

in relation table routines.

f_articlefieldname

```
f_articlefieldname(a_key number)
return varchar
```

Get the name (column c_name) of an article field. This is a convenience function to avoid joins in relation table routines.

f_articleclassname

```
f_articleclassname(a_key number)
return varchar
```

Get the full name of an article class. A slash is used to separate parts of the name.

Example:

```
select f_articleclassname(c_key)
from t_articleclasses
order by 1
```

Output:

```
apples
apples/golden
apples/golden/large
apples/golden/small
apples/granny
bananes
bananes/chiquita
bananes/havelar
oranges
...
```

f_articlehasattribute

```
f_articlehasattribute (a_articlekey number, a_attributename varchar)
return number
```

Returns 1 if the article with the given key has the named attribute, and 0 otherwise.

f_articleid

```
f_articleid(a_key number)
return varchar
```

Get the ID (column c_id) of an article.

f_articleidname

```
f_articleidname(a_key number)
return varchar
```

Get the ID and name (c_id || ' ' || c_name) of an article. This is a convenience function to avoid joins in relation table routines.

f_articlename

```
f_articlename(a_key number)
return varchar
```

Get the name (column c_name) of an article. This is a convenience function to avoid joins in relation table routines.

f_articleordering

```
f_articleordering(a_key number)
return varchar
```

Get the lowercased ID path (concatenated c_id columns) of an article, where each ID is padded with blanks to 35 characters. This function is typically used in order clauses.

f_articles_propagates

```
f_articles_propagates(a_key number, a_item varchar)
return boolean
```

Check if propagating of a column/options table is enabled in a certain article. The main usage is in the propagate triggers.

Example:

```
# check if article with key 123 propagates the column c_name
if f_articles_propagates(123, 'c_name') then
  #do something
end if;
```

f_auditstatus

```
f_auditstatus(a_name varchar)
return number
```

Get audit status for the standard table with the given base name (omit the prefix `t_`).

For example `f_auditstatus('orderitems')` will get the audit status for table `t_orderitems`.

Return:

0 = Audit is disabled
1 = Audit is enabled

f_articletreeid

```
f_articletreeid(a_key number)
return varchar
```

Get the tree ID's (column c_id) of an article as a path.

f_availablestock

```
f_availablestock(a_key number, a_unit varchar)
return number
```

Get the available stock in the desired order unit for a given article key.

Stocks on locations with status *open* and *drawing* are added, and the missing (not yet picked) units in open trips where status is not *history* are subtracted to get the quantity.

Order units: 'L' = LU, 'T' = TU, 'C' = CU

f_availabletripstock

```
f_availabletripstock(a_ordertripkey number, a_articlekey number, a_unit
varchar)
return number
```

Get the stock available for picking in the desired order unit for a given combination of order trip and article key.

Stocks on locations with status *open* and *drawing* are added, and the missing (not yet picked) units are subtracted to get the quantity. The missing quantity is calculated of all preceeding, unarchived order trips. „Preceding“ are all order trips displayed before the given trip in the tree view of the order trips application, e.g. where order trips are ordered by t_ordertrip.c_date, t_trip.c_id, t_ordertrip.c_key.

Order units: 'L' = LU, 'T' = TU, 'C' = CU

f_consumerprice

```
f_consumerprice(a_articlekey number, a_datetime timestamp)
return number
```

Returns the consumer price of an article for the given date and time.

Returns null if there is no consumer price.

f_currentstock

```
f_currentstock(a_key number, a_unit varchar)
return number
```

Get the current stock in the desired order unit for a given article key.

Stocks on locations with status *open* and *drawing* are included in the sum. The sum is rounded to the next lower integer value.

Order units: 'L' = LU, 'T' = TU, 'C' = CU

f_deliverpickstatus

```
f_deliverpickstatus()
return number
```

Get status of pick delivery registration executed by p_deliverpicks.

f_departmentaccess

```
f_departmentaccess(a_departmentkey number, a_userkey number)
return number
```

Check access right of a department/user combination. Returns 1 if access is granted, 0 for no access.

f_departmentid

```
f_departmentname(a_key number)
return varchar
```

Get the id (column c_id) of a department. This is a convenience function to avoid joins in relation table routines.

f_departmentidname

```
f_departmentidname(a_key number)
return varchar
```

Get the id and name (c_id||' '||c_name) of a department. This is a convenience function to avoid joins in relation table routines.

f_departmentname

```
f_departmentname(a_key number)
return varchar
```

Get the name (column c_name) of a department. This is a convenience function to avoid joins in relation table routines.

f_findstocklocation

```
f_findstocklocation(a_path varchar)
return number
```

Find the stock location key from a given stock location path. Returns the key if found, or null if no such stock location is found.

This is the inverse of function f_stocklocationid.

f_haspermission

```
f_haspermission(a_permission varchar, a_userkey number)
return number
```

Check function access right of a user. Returns 1 if access is granted, 0 for no access.

Example:

```
# check if user with key 12 has permission to add articles
if f_haspermission('articles.add', 12) then
  #do something
end if;
```

f_inarticleattributes

```
f_inarticleattributes(a_article number, a_attribute number)
return number
```

Check if the combination of article and attribute exists in table t_articleattributes. Returns 1 when true, 0 when false.

f_inarticleclass

```
f_inarticleclass(a_testkey number, a_classkey number)
return number
```

Check if a_testkey is equal to a_classkey, or is a child class of a_classkey. Returns 1 when true, 0 when false.

f_inmergeclass

```
f_inmergeclass(a_testkey number, a_classkey number)
return number
```

Check if a_testkey is equal to a_classkey, or is a child class of a_classkey. Returns 1 when true, 0 when false.

f_inpartnerattributes

```
f_inpartnerattributes(a_partner number, a_attribute number)
return number
```

Check if the combination of partner and attribute exists in table t_partnerattributes. Returns 1 when true, 0 when false.

f_inpartnerclass

```
f_inpartnerclass(a_testkey number, a_classkey number)
return number
```

Check if a_testkey is equal to a_classkey, or is a child class of a_classkey. Returns 1 when true, 0 when false.

f_inpartnerdepartments

```
f_inpartnerdepartnments(a_partner number, a_department number)
return number
```

Check if the combination of partner and department exists in table t_partnerdepartments. Returns 1 when true, 0 when false.

f_prodclass

```
f_prodclass(a_testkey number, a_classkey number)
return number
```

Check if a_testkey is equal to a_classkey, or is a child class of a_classkey. Returns 1 when true, 0 when false.

f_inpromotion

```
f_inpromotion(a_key number, a_datetime timestamp)
return number
```

Check if the article with the given key has a promotion at the given date and time.

Returns 1 if it is in promotion, 0 if no promotion.

f_insalechannel

```
f_insalechannel(a_testkey number, a_classkey number)
return number
```

Check if a_testkey is equal to a_classkey, or is a child class of a_classkey. Returns 1 when true, 0 when false.

f_insaleclass

```
f_insaleclass(a_testkey number, a_classkey number)
return number
```

Check if a_testkey is equal to a_classkey, or is a child class of a_classkey. Returns 1 when true, 0 when false.

f_inshipclass

```
f_inshipclass(a_testkey number, a_classkey number)
return number
```

Check if a_testkey is equal to a_classkey, or is a child class of a_classkey. Returns 1 when true, 0 when false.

f_instockattributes

```
f_instockattributes(a_stocklocation number, a_attribute number)
return number
```

Check if the combination of stock location and attribute exists in table t_stockattributes. Returns 1 when true, 0 when false.

f_intripclass

```
f_intripclass(a_testkey number, a_classkey number)
return number
```

Check if a_testkey is equal to a_classkey, or is a child class of a_classkey. Returns 1 when true, 0 when false.

f_isadmin

```
f_isadmin(a_username varchar)
return number
```

Check if the user with the given database user name has the necessary DB privileges to act as user admin.

For Oracle, the required system privileges are CREATE USER, ALTER USER, DROP USER, SELECT ANY DICTIONARY, GRANT ANY PRIVILEGE and GRANT ANY ROLE.

For PostgreSQL the required role privileges are rolcreaterole and rolcanlogin.

Returns 1 when having the required privilegs, 0 when not sufficient privilegs.

f_isdba

```
f_isdba(a_username varchar)
return number
```

Check if the user with the given database user name has DBA privileges.

For Oracle, the required role is DBA.

For PostgreSQL the required role privileges are rolsuper and rolcanlogin.

Returns 1 when having DBA privilegs, 0 when not.

f_key

```
f_key(a_key varchar)
```

```
return number
```

Returns the value of a_key in case it is not null, or -1 if it is null. This makes joining of keys, which could be null, more convenient because you need not care about nulls in the clause.

Example:

```
select c_key from t_settings
where f_key(c_parent)=f_key(?)
and f_key(c_user)=f_key(?)
and lower(c_name)=lower(?)
and c_type='g'
```

f_keylist

```
f_keylist(a_keys in keytable)
return text
```

Get a comma separated list from a key table.

f_keytable

```
f_keytable((a_list in text))
return keytable
```

Get a key table from a comma separated list.

f_languagename

```
f_languagename(a_key number)
return varchar
```

Get the name (column c_name) of a language. This is a convenience function to avoid joins in relation table routines.

f_licensesinuse

```
f_licensesinuse(a_licenser varchar, a_licgroup varchar)
return number
```

Get the number of licenses in use for a certain license group.

Example:

```
select f_licensesinuse('IBK Software AG', 'Master Data')
```

f_mergeclassname

```
f_mergeclassname(a_key number)
return varchar
```

Get the full name of a merge class. A slash is used to separate parts of the name.

Example:

```
select f_mergeclassname(c_key)
from t_mergeclasses
order by 1
```

f_methodidname

```
f_methodidname(a_key number)
return varchar
```

Get the id and name (c_id||' '||c_name) of a production method. This is a convenience function to avoid joins in relation table routines.

f_nextkey

```
f_nextkey(a_name varchar)
return number
```

Get the next key for a table with name t_||a_name. The key is generated from the sequence s_||a_name. The function makes sure that a key yet unused in the table is returned.

Example:

```
insert into t_clients(c_key, c_name)
values (f_nextkey('clients'), 'New customer')
```

f_nextval

```
f_nextval(a_sequence varchar)
return number
```

Get the next value of a sequence in a database independent way.

Example:

```
insert into t_clients(c_key, c_name)
values (f_nextval('s_clients'), ?)
```

To make sure that the key is yet unused in the target table, use function f_nextkey instead.

f_ordercollocked

```
f_ordercollocked(a_key varchar)
return number
```

Checks if an order column, the trip it belongs to, or any of its items is locked. Also checks if any rows are locked belonging to the items of this column.

Returns 1 if any locks are present, and 0 if no locks are present.

f_ordercolslocked

```
f_ordercolslocked(a_keys keytable)
return number
```

Checks if any of a list of order columns, the trip they belong to, or any of theire items ars locked. Also checks if any rows are locked belonging to the items.

Returns 1 if any locks are present, and 0 if no locks are present.

f_ordercolsremarks

```
f_ordercolsremarks(a_keys in keytable, a_sep in varchar)
return text
```

Get joined remarks of of a list of order columns. a_sep is used as separator between the remarks.

f_ordercolsusers

```
f_ordercolsusers(a_keys in t_keytable, a_zone in number, a_sep in varchar)
return varchar
```

Get all users holding locks on a list of order columns with items in the given zone.

The separator text (for example a newline char or a comma) is given in c_sep.

f_orderitemlocker

```
f_orderitemlocker(a_key varchar)
return number
```

Returns key of the actor holding a lock on this order item, or NULL if no lock is present.

f_orderitemstatus

```
f_orderitemstatus(a_key varchar)
return varchar
```

Returns the status or the order item as:

Return	Status	Description
'o'	open	No picks are yet present.
'p'	picked	Picks are present but some or all have c_delivery = NULL.
'd'	delivered	Picks are present, all have c_delivery != NULL but some or all have c_invoice = NULL
'i'	invoiced	Picks are present, all have c_delivery != NULL and c_invoice != NULL

f_orderitemtripstatus

```
f_orderitemtripstatus(a_orderitemkey varchar)
return varchar
```

Returns the status or the order trip where the order item belongs to.

f_orderrowlocked

```
f_orderrowlocked(a_key varchar)
return number
```

Checks if an order row, the trip it belongs to, or any of its items is locked. Also checks if any columns are locked belonging to the items of this row.

Returns 1 if any locks are present, and 0 if no locks are present.

f_orderrowslocked

```
f_orderrowslocked(a_keys keytable)
return number
```

Checks if any of a list of order rows, the trip they belong to, or any of theire items ars locked. Also checks if any columns are locked belonging to the items.

Returns 1 if any locks are present, and 0 if no locks are present.

f_orderrowsremarks

```
f_orderrowsremarks(a_keys in keytable, a_sep in varchar)
```

```
return text
```

Get joined remarks of of a list of order rows. a_sep is used as separator between the remarks.

f_orderrowsusers

```
f_orderrowsusers(a_keys in t_keytable, a_zone in number, a_sep in varchar)
return varchar
```

Get all users holding locks on a list of order rows with items of the given zone.

The separator text (for example a newline char or a comma) is given in c_sep.

f_ordertriplocked

```
f_ordertriplocked(a_key varchar)
return number
```

Checks if an open trip or any of its rows, columns or items is locked.

Returns 1 if any locks are present, and 0 if no locks are present.

f_originidname

```
f_originidname(a_key number)
return varchar
```

Get the id and name (c_id||' '||c_name) of an origin. This is a convenience function to avoid joins in relation table routines.

f_packageidname

```
f_packageidname(a_key number)
return varchar
```

Get the ID and name (c_id || ' ' || c_name) of a package. This is a convenience function to avoid joins in relation table routines.

f_packageidnamekg

```
f_packageidnamekg(a_key number)
return varchar
```

Get the ID, name and weight in kg of a package, for example `0101.305 Migros-A 1.850 kg`.

f_packagenamekg

```
f_packagenamekg(a_key number)
return varchar
```

Get the name (or ID if name is null) and weight in kg of a package, for example `Migros-A 1.850` kg.

f_packjoblocked

```
f_packjoblocked(a_key varchar, a_alllocks in number)
return number
```

Checks if a packing job or the production it belongs to is locked.

If a_alllocks is 0, non-exclusive packing job locks are ignored.
If a_alllocks is 1, absolutely no lock shall be present.

Returns 1 if locks are present, and 0 if no locks are present.

f_packjobusers

```
f_packjobusers(a_keys in keytable, a_sep in varchar)
return varchar
```

```
Get all users holding locks on a list of packing jobs.
The separator text (for example a newline char or a comma) is given in
c_sep.
```

f_packprodlocked

```
f_packprodlocked(a_key varchar)
return number
```

Checks if a packing production or any of jobs is locked.

Returns 1 if any locks are present, and 0 if no locks are present.

f_partneraccess

```
f_partneraccess(a_partnerkey number, a_userkey number)
return number
```

Check access right of a partner/user combination. Returns 1 if access is granted indirectly via department, 0 for no access.

f_partnerclassname

```
f_partnerclassname(a_key number)
return varchar
```

Get the full name of a partner class. A slash is used to separate parts of the name.

Example:

```
select f_partnerclassname(c_key)
from t_partnerclasses
order by 1
```

f_partnerhasattribute

```
f_partnerhasattribute(a_partnerkey number, a_attributename varchar)
return number
```

Returns 1 if the partner with the given key has the named attribute, and 0 otherwise.

f_partnerid

```
f_partnerid(a_key number)
return varchar
```

Get the ID path (column c_id) of a partner.

f_partneridname

```
f_partneridname(a_key number)
return varchar
```

Get the ID and name (c_id||' '||c_name) of a partner. This is a convenience function to avoid joins in relation table routines.

f_partnername

```
f_partnername(a_key number)
return varchar
```

Get the name (column c_name) of a partner. This is a convenience function to avoid joins in relation table routines.

f_partnerordering

```
f_partnerordering(a_key number)
return varchar
```

Get the lowercased ID path (concatenated c_id columns) of a partner, where each ID is padded with blanks to 35 characters. This function is typically used in order clauses.

f_partners_propagates

```
f_partners_propagates(a_key number, a_item varchar)
return boolean
```

Check if propagating of a column/options table is enabled in a certain partner. The main usage is in the propagate triggers.

Example:

```
# check if partner with key 123 propagates the column c_name
if f_partners_propagates(123, 'c_name') then
  #do something
end if;
```

f_partnertreeid

```
f_partnertreeid(a_key number)
return varchar
```

Get the tree ID's (column c_id) of a partner as a path.

f_pickartlocked

```
f_pickartlocked(a_ordertrip in number,
                a_article in number,
                a_cpr in number,
                a_prcd in varchar,
                a_tu_lu in number,
                a_cu_tu in number,
                a_pu_cu in number,
                a_ou in varchar,
                a_zone in number)
return number
```

Check if orders with the given column parameters are locked. All levels are checked: trip, column, row and item locks.

Returns 1 if any locks are present, and 0 if no locks are present.

f_pickartlocked1

```
f_pickartlocked1(a_ordertrip in number,
                 a_article in number,
                 a_cpr in number,
                 a_prcd in varchar,
                 a_tu_lu in number,
                 a_cu_tu in number,
                 a_pu_cu in number,
                 a_ou in varchar)
return number
```

Check if orders with the given column parameters are locked. All levels are checked: trip, column, row and item locks.

Returns 1 if any locks are present, and 0 if no locks are present.

f_pickartlocked2

```
f_pickartlocked2(a_ordertrip in number,
                 a_article in number,
                 a_cpr in number,
                 a_prcd in varchar,
                 a_tu_lu in number,
                 a_cu_tu in number,
                 a_pu_cu in number,
                 a_ou in varchar,
                 a_selldays in number,
                 a_expiredays in number,
                 a_zone in number)
return number
```

Check if orders with the given column parameters are locked. All levels are checked: trip, column, row and item locks.

Returns 1 if any locks are present, and 0 if no locks are present.

f_pickartlocked3

```
f_pickartlocked3(a_ordertrip in number,
                 a_article in number,
                 a_tu_lu in number,
                 a_cu_tu in number,
                 a_pu_cu in number,
                 a_ou in varchar,
```

```
                  a_zone in number)
return number
```

Check if orders with the given column parameters are locked. All levels are checked: trip, column, row and item locks.

Returns 1 if any locks are present, and 0 if no locks are present.

f_pickcolzonelocked

```
f_pickcolzonelocked(a_ordercol in number, a_zone in number)
return number
```

Checks if any order column with items in the given zone is locked.

Returns 1 if any locks are present, and 0 if no locks are present.

f_pickpnrlockers

```
f_pickpnrlockers(a_ordertrip in number, a_partner in number, a_zone in
number, a_mergeclass in number)
return numbers
```

Get a list of keys of the the actors holding locks on the items of a distinct ordertrip key, partner key, zone key and merge class key. All levels are checked, so also trip locks or column locks belonging to the items are listed.

The return type is t_keytable (table of number) for oracle, and setof numeric for PostgreSQL.

Oracle example:

```
select distinct u.c_name, a.c_nodename"
from table(f_pickpnrlockers(1,2,3,4)) l"
inner join t_actors a on l.column_value=a.c_key"
inner join t_users u on a.c_user=u.c_key"
order by 1,2
```

PostgreSQL example:

```
select distinct u.c_name, a.c_nodename"
from f_pickpnrlockers(1,2,3,4) l"
inner join t_actors a on l=a.c_key"
inner join t_users u on a.c_user=u.c_key"
order by 1,2
```

f_pickrowlockers

```
f_pickrowlockers(a_orderrow in number, a_zone in number, a_mergeclass in
number)
return numbers
```

Get a list of keys of the the actors holding locks on the items of a distinct orderrow key, picking zone key and merge class key. All levels are checked, so also trip locks or column locks belonging to the items are listed.

The return type is t_keytable (table of number) for oracle, and setof numeric for PostgreSQL.

Oracle example:

```
select distinct u.c_name, a.c_nodename"
from table(f_pickrowlockers(1,2,3)) l"
inner join t_actors a on l.column_value=a.c_key"
inner join t_users u on a.c_user=u.c_key"
order by 1,2
```

PostgreSQL example:

```
select distinct u.c_name, a.c_nodename"
from f_pickrowlockers(1,2,3) l"
inner join t_actors a on l=a.c_key"
inner join t_users u on a.c_user=u.c_key"
order by 1,2
```

f_pkgattributename

```
f_pkgattributename(a_key number)
return varchar
```

Get the name (column c_name) of a package attribute. This is a convenience function to avoid joins in relation table routines.

f_pricecode

```
f_pricecode(a_articlekey number, a_datetime timestamp)
return varchar
```

Returns the price code for a article at the given date and time:

n = regular price
p = promotion
i = introduction
s = sellout

Returns null if there is no price.

f_pricekey

```
f_pricekey(a_articlekey number, a_datetime timestamp)
return number
```

Returns the key referencing t_articleprices for a article at the given date and time.

Returns null if there is no price.

f_prodaccess

```
f_prodaccess(a_prodkey number, a_userkey number)
return number
```

Check access right of a production/user combination. Returns 1 if access is granted, 0 for no access.

f_prodclassname

```
f_prodclassname(a_key number)
return varchar
```

Get the full name of a production class. A slash is used to separate parts of the name.

Example:

```
select f_prodclassname(c_key)
from t_prodclasses
order by 1
```

f_prodidname

```
f_prodidname(a_key number)
return varchar
```

Get the ID and name (c_id||' '||c_name) of a production. This is a convenience function to avoid joins in relation table routines.

f_prodname

```
f_prodname(a_key number)
return varchar
```

Get the name (column c_name) of a production. This is a convenience function to avoid joins in relation table routines.

f_produceridname

```
f_produceridname(a_key number)
return varchar
```

Get the ID and name (c_id||' '||c_name) of a producer. This is a convenience function to avoid joins in relation table routines.

f_purchaseitemlocker

```
f_purchaseitemlocker(a_key varchar)
return number
```

Returns key of the actor holding a lock on this purchase item, or NULL if no lock is present.

f_purchaseorderlocked

```
f_purchaseorderlocked(a_key varchar)
return number
```

Checks if a purchase order or any of its items is locked.

Returns 1 if any locks are present, and 0 if no locks are present.

f_reportaccess

```
f_reportaccess(a_reportkey number, a_userkey number)
return number
```

Check access right of a report/user combination.

Returns:

0 = no access
1 = read only
2 = read and write

f_reportpath

```
f_reportpath(a_key in number)
return varchar
```

Get the full path name of a report. The slash is used to separate parts of the path, similar to file system paths on linux.

Example:

```
select f_reportpath(c_key), c_type from t_reports
```

f_resourceid

```
f_resourceid(a_key number)
return varchar
```

Get the id (column c_id) of a resource. This is a convenience function to avoid joins in relation table routines.

f_resourceidname

```
f_resourceidname(a_key number)
return varchar
```

Get the id and name (c_id||' '||c_name) of a resource. This is a convenience function to avoid joins in relation table routines.

f_resourcename

```
f_resourcename(a_key number)
return varchar
```

Get the name (column c_name) of a resource. This is a convenience function to avoid joins in relation table routines.

f_salechannelname

```
f_salechannelname(a_key number)
return varchar
```

Get the full name of a sale channel. A slash is used to separate parts of the name.

Example:

```
select f_salechannelname(c_key)
from t_f_salechannels
order by 1
```

f_saleclassname

```
f_saleclassname(a_key number)
```

```
return varchar
```

Get the full name of a sale class. A slash is used to separate parts of the name.

Example:

```
select f_saleclassname(c_key)
from t_saleclasses
order by 1
```

f_scheduleinfo

```
f_scheduleinfo(a_schedule varchar, a_info varchar, a_index number)
return number
```

Extract information from a schedule field.

The following parts can be extracted:

Info	**Index**	**Return**
minute	0 ... 59	1 if the minute is included, 0 if not
hour	0 ... 59	1 if the hour is included, 0 if not
dayofmonth	1 ... 31	1 if the day of month is included, 0 if not
weekday	0 ... 7, where 0/7 = sunday, 1 = monday, etc.	1 if the weekday is included, 0 if not
month	1 ... 12	1 if the month is included, 0 if not
firsttime	null	The first schedule time as hour.minute
lasttime	null	The last schedule time as hour.minute

Example:

```
select f_scheduleinfo('* 7 * mon-fri *', 'weekday', 2) from t_dual
// returns 1 because tuesday is included

select f_scheduleinfo('15,45 7-18 * mon-fri *', 'firsttime', null) from
t_dual
// returns 7.15

select f_scheduleinfo('15,45 7-18 * mon-fri *', 'lasttime', null) from
t_dual
// returns 18.45
```

f_sessionid

```
f_sessionid()
return number
```

Get the session id in a database independant manner.

Example:

```
select f_sessionid() from t_dual
```

f_shipclassname

```
f_shipclassname(a_key number)
return varchar
```

Get the full name of a shipping class. A slash is used to separate parts of the name.

Example:

```
select f_shipclassname(c_key)
from t_shipclasses
order by 1
```

f_ssccprefix

```
f_ssccprefix()
return varchar
```

Returns the SSCC prefix from the centrals system settings in `Labeling/LU/SSCC Prefix`

f_stkattributename

```
f_stkattributename(a_key number)
return varchar
```

Get the name (column c_name) of a stock attribute. This is a convenience function to avoid joins in relation table routines.

f_stocklocationid

```
f_stocklocationid(a_key number)
return varchar
```

Get the tree path (column c_id) of a stock location. A slash is used to separate parts of the path.

f_stocklocationoccupancy

```
f_stocklocationoccupancy(a_key varchar)
return varchar
```

Check the occupancy of a stock location.

Returns the state as:

Return	Status	Description
'e'	empty	No stocks are assigned to the location.
'o'	occupied	One or more stocks is/are assigned to the location. The stock may however be 0, or even below zero (for virtual stocks).

f_stocklocationordering

```
f_stocklocationordering(a_key number)
return varchar
```

Get the lowercased ID path (concatenated c_id columns) of a stock location, where each ID is padded with blanks to 35 characters. This function is typically used in order clauses.

f_stocklocations_propagates

```
f_stocklocations_propagates(a_key number, a_item varchar)
return boolean
```

Check if propagating of a column/options table is enabled in a certain stock location. The main usage is in the propagate triggers.

Example:

```
# check if stock location with key 123 propagates the column c_status
if f_stocklocations_propagates(123, 'c_status') then
  #do something
end if;
```

f_stockreasonname

```
f_stockreasonname(a_key number)
return varchar
```

Get the name (column c_name) of a stock reason. This is a convenience function to avoid joins in relation table routines.

f_textline

```
f_textline (a_text varchar, a_line number)
return varchar
```

Extract a distinct line from a multiline text. The line breaks are assumed to be CR or CR/LF. The line

numbering starts at 1.

Example:

```
# Get first 2 lines of label text:
select
  f_textline(c_labeltext,1) as line1,
  f_textline(c_labeltext,2) as line2
from t_partners;
```

f_thisactor

```
f_thisactor()
return number
```

Get the actor key (t_actors.c_key) of the current session.

Works only if checked in to the datastore, and returns NULL otherwise.

f_thisuser

```
f_thisuser()
return number
```

Get the user key (t_users.c_key) of the current session.

Works only if checked in to the datastore, and returns NULL otherwise.

f_tripaccess

```
f_tripaccess(a_tripkey number, a_userkey number)
return number
```

Check access right of a trip/user combination. Returns 1 if access is granted, 0 for no access.

f_tripclassname

```
f_tripclassname(a_key number)
return varchar
```

Get the full name of a trip class. A slash is used to separate parts of the name.

Example:

```
select f_tripclassname(c_key)
from t_tripclasses
```

```
order by 1
```

f_tripidname

```
f_tripidname(a_key number)
return varchar
```

Get the ID and name (c_id||' '||c_name) of a trip. This is a convenience function to avoid joins in relation table routines.

f_tripname

```
f_tripname(a_key number)
return varchar
```

Get the name (column c_name) of a trip. This is a convenience function to avoid joins in relation table routines.

f_usernameid

```
f_usernameid(a_name varchar, a_userid number)
return varchar
```

- Returns a_userid enclosed in square brackets if a_name is null.
- Returns a_name followed by a blank and a_userid in square brackets otherwise.

Examples:

```
f_usernameid(null, 10)  /* [10] */
```

```
f_usernameid('John', 20) /* John [20] */
```

```
select f_usernameid(c_name, c_userid) from t_users
```

f_vatname

```
f_vatname(a_key number)
return varchar
```

Get the name (column c_name) of a value added tax. This is a convenience function to avoid joins in relation table routines.

f_zoneaccess

```
f_zoneaccess(a_zonekey number, a_userkey number)
return number
```

Check access right of a picking zone/user combination. Returns 1 if access is granted, 0 for no access.

f_zonename

```
f_zonename(a_key number)
return varchar
```

Get the name of a picking zone.

f_zonepartnerordering

```
f_zonepartnerordering(a_zone number, a_partner number)
return varchar
```

Get the partner ordering within a zone as a string. This function is typically used in order clauses.

Procedure Reference

p_deliverpicks

```
p_deliverpicks(
  a_sscc in varchar2,       /* SSCC of the aggregated parent object. */
  a_grailist in t_strtable, /* GRAI's of the aggregated child objects. */
  a_timeout in number,      /* Time out setting when orders are locked. */
  a_check in number         /* Validation flags. */
)
```

> The procedure is only available in Oracle.

Aggregate a number of GRAI's into one SSCC, and set the affected picks to *delivered*.

- The procedure searches in the *active* order trips to find picks with any of the given GRAI's.
- On all found picks the validations defined in `a_check` are performed.
- Not yet completed picks (where `c_delivery` was `NULL` before) are updated as:
  - `c_sscc` = `a_sscc`
  - `c_delivery` = New delivery ID for each procedure call
  - `c_delivby` = Current user executing the procedure
  - `c_delivdate` = Currrent timestamp
- Picks completed earlier (where `c_delivery` was `NOT NULL` before) stay unchanged.

Parameter a_timeout

Value	Effect
< 0	Wait forever for the lock, never raise ORA-20002.
0	Do not wait, raise ORA-20002 immediately if orders are locked by someone else.
> 0	Wait for the given number of seconds max before raising ORA-20002.

Parameter a_check

Value	Check
1	Check for unique SSCC (if `t_orderpicks.c_sscc` was `NOT NULL` before).
2	Check for unique partner in all orders (`t_orderrows.c_partner`).
4	Check for unique order trip in all orders (`t_ordertrips.c_key`).

The validation flags can be added to enable multiple checks. For example to check for unique SSCC and for unique partner, set 1+2

Example

Aggregate 3 GRAI's and finish the picks. Wait maximum 10 seconds for lock. Apply all validations:

```
begin
  p_deliverpicks(
    '7613264.1234567890',
    t_strtable(
      '7613264.00307.100005001118',
      '7613264.00307.100005001119',
      '7613264.00307.100005001122'
    ),
    10, 1+2+4
  );
end;
/
```

The procedure raises following errors in case of problems:

Error	Message / ***Remarks***
ORA-20001	Invalid user. Caller must be a BPS user defined in table t_users.
ORA-20002	Lock attempt timed out. Affected order objects were in use too long.
ORA-20003	Mixed SSCC's in same aggregation. (1) *One or more of the picks already had an other SSCC.*
ORA-20004	Mixed partners in same aggregation. (1) *The picks belong to more than one partner.*
ORA-20005	Mixed order trips in same aggregation. (1) *The picks belong to more than one order trip.*

(1) Key and GRAI of the pick where the problem was detected are appended to the message.

p_deliverssccpicks

```
p_deliverssccpicks(
  a_sscc in varchar2,       /* SSCC of the aggregated objects. */
  a_timeout in number,      /* Time out setting when orders are locked. */
  a_check in number         /* Validation flags. */
)
```

> The procedure is only available in Oracle.

Aggregate a SSCC and set the affected picks to *delivered*.

- The procedure searches in the *active* order trips to find picks with any of the given SSCC.
- On all found picks the validations defined in `a_check` are performed.
- Not yet completed picks (where `c_delivery` was `NULL` before) are updated as:
  - `c_delivery` = New delivery ID for each procedure call
  - `c_delivby` = Current user executing the procedure
  - `c_delivdate` = Currrent timestamp
- Picks completed earlier (where `c_delivery` was `NOT NULL` before) stay unchanged.

Parameter a_timeout

Value	Effect
< 0	Wait forever for the lock, never raise ORA-20002.
0	Do not wait, raise ORA-20002 immediately if orders are locked by someone else.
> 0	Wait for the given number of seconds max before raising ORA-20002.

Parameter a_check

Value	Check
2	Check for unique partner in all orders (`t_orderrows.c_partner`).
4	Check for unique order trip in all orders (`t_ordertrips.c_key`).

The validation flags can be added to enable multiple checks. For example to check for unique partner and for unique ordertrip, set 2+4

Example

Aggregate SSCC and finish the picks. Wait maximum 10 seconds for lock. Apply all validations:

```
begin
  p_deliverssccpicks(
    '7613264.1234567890',
    10, 2+4
  );
end;
/
```

The procedure raises following errors in case of problems:

Error	**Message / *Remarks***
ORA-20001	Invalid user. Caller must be a BPS user defined in table t_users.
ORA-20002	Lock attempt timed out. Affected order objects were in use too long.
ORA-20004	Mixed partners in same aggregation. (1) *The picks belong to more than one partner.*
ORA-20005	Mixed order trips in same aggregation. (1) *The picks belong to more than one order trip.*

(1) Key and SSCC of the pick where the problem was detected are appended to the message.

p_disabletriggers

```
p_disabletriggers()
```

Disables all database triggers of the current database user/schema.

Since PostgreSQL does not have procedures, it is implemented as function without return value.

Oracle example:

```
begin p_disabletriggers(); end;
```

PostgreSQL example:

```
select p_disabletriggers()
```

p_enabletriggers

```
p_enabletriggers()
```

Enables all database triggers of the current database user/schema.

Since PostgreSQL does not have procedures, it is implemented as function without return value.

Oracle example:

```
begin p_enabletriggers(); end;
```

PostgreSQL example:

```
select p_enabletriggers()
```

p_fixchars

```
p_fixchars()
```

Changes all VARCHAR2 char sizes from „BYTE“ to „CHAR“ in Oracle.

Background: Some very old BPS2 databases still have objects defined in BYTE units, while BPS generally forces CHAR in new installations. The procedure is executed automatically by database update scripts.

Example:

```
begin p_fixchars(); end;
```

p_purgeactors

```
p_purgeactors()
```

Removes any actors for which the database session no longer exists.

Since PostgreSQL does not have procedures, it is implemented as function without return value.

Oracle example:

```
begin p_purgeactors(); end;
```

PostgreSQL example:

```
select p_purgeactors()
```

p_purgeemptyorders

```
p_purgeemptyorders()
```

Removes any empty order trips (table t_ordertrips), rows (table t_orderrows) and columns (table t_ordercols). Typically this procedure will be executed by applications after deleting any order columns, rows or items to clean up.

Since PostgreSQL does not have procedures, it is implemented as function without return value.

Oracle example:

```
begin p_purgeemptyorders(); end;
```

PostgreSQL example:

```
select f_purgeemptyorders()
```

p_purgeemptytriporders

```
p_purgeemptytriporders(a_key in number)
```

Removes any empty order trips (table t_ordertrips), rows (table t_orderrows) and columns (table t_ordercols) of a distinct trip. Typically this procedure will be executed by applications after deleting any order columns, rows or items to clean up.

Since PostgreSQL does not have procedures, it is implemented as function without return value.

Oracle example:

```
begin p_purgeemptytriporders(1234); end;
```

PostgreSQL example:

```
select p_purgeemptytriporders(1234)
```

p_rebuildordershadow

```
p_rebuildordershadow()
```

Rebuilds the order shadow columns in t_orderitems, t_ordercols and t_orderrows. The order shadow is made up by replication of redundant values and sums that are present for performance reasons, and are updated automatically by triggers. The shadow could get out of sync when triggers are disabled or somebody accidently updates a shadow column. In this dase, the procedure will fix the issue. Depending on the amount of orders in the database, the execution time can be very long.

Since PostgreSQL does not have procedures, it is implemented as function without return value.

Oracle example:

```
begin p_rebuildordershadow(); end;
```

PostgreSQL example:

```
select p_rebuildordershadow()
```

p_rebuildpackshadow

```
p_rebuildpackshadow()
```

Rebuilds the packing record shadow columns in t_packjobs. The packing shadow is made up by replication of redundant values and sums that are present for performance reasons, and are updated automatically by triggers. The shadow could get out of sync when triggers are disabled or somebody accidently updates a shadow column. In this dase, the procedure will fix the issue. Depending on the amount of packing jobs in the database, the execution time can be very long.

Since PostgreSQL does not have procedures, it is implemented as function without return value.

Oracle example:

```
begin p_rebuildpackshadow(); end;
```

PostgreSQL example:

```
select p_rebuildpackshadow()
```

p_rebuildstockshadow

```
p_rebuildstockshadow()
```

Rebuilds the stock shadow columns in t_articles and t_stock. The order shadow is made up by replication of redundant values and sums that are present for performance reasons, and are updated automatically by triggers. The shadow could get out of sync when triggers are disabled or somebody accidently updates a shadow column. In this dase, the procedure will fix the issue.

Since PostgreSQL does not have procedures, it is implemented as function without return value.

Oracle example:

```
begin p_rebuildstockshadow(); end;
```

PostgreSQL example:

```
select p_rebuildstockshadow()
```

p_sleep

```
p_sleep(a_millisecs in number)
```

Delays SQL execution for the given time in milliseconds.

Since PostgreSQL does not have procedures, it is implemented as function without return value.

Oracle example:

```
begin p_sleep(3000); end;
```

PostgreSQL example:

```
select p_sleep(3000)
```