UF_UGMGR_add_product_assembly_part (view source)

Defined in: uf_ugmgr.h

Overview

Adds a Product Assembly to the list of those recognised by the current NX Manager session.

Specify the Product Assembly in CLI format.

Environment

Internal and External

History

Originally released in NX 5.0 and is mandatory if Longer IDs functionality is enabled NX/Manager

Required License(s)

gateway

```
int UF_UGMGR_add_product_assembly_part
(
    const char * product
)
```

const char *	**product**	Input	Name of the product assembly in CLI form.

UF_UGMGR_add_to_folder (view source)

Defined in: uf_ugmgr.h

Overview

Adds the specified object to the specified folder. You may add only folders or parts to the specified folder.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_add_to_folder
(
    UF_UGMGR_tag_t object_to_add,
    UF_UGMGR_tag_t folder
)
```

UF_UGMGR_tag_t	**object_to_add**	Input	Database tag of the object to add to the specified folder.
UF_UGMGR_tag_t	**folder**	Input	Database tag of the folder in which to place the object.

UF_UGMGR_ask_autolock_status (view source)

Defined in: uf_ugmgr.h

Overview
Allows querying of the autolock status for checking out of parts when modified.

Environment
Internal and External

See Also
UF_UGMGR_set_autolock_status

History
This function was originally released in V16.0.2

Required License(s)
gateway

```
int UF_UGMGR_ask_autolock_status
(
    logical * current_value
)
```

logical *	**current_value**	Output	Current value of the autolock status

UF_UGMGR_ask_config_rule (view source)

Defined in: uf_ugmgr.h

Overview
Returns your current configuration rule.

Environment
Internal and External

Required License(s)
gateway

```
int UF_UGMGR_ask_config_rule
(
    char current_rule [ UF_UGMGR_NAME_BUFSIZE ]
)
```

char	**current_rule [UF_UGMGR_NAME_BUFSIZE]**	Output	Your current configuration rule. Declare this parameter with array size UF_UGMGR_NAME_BUFSIZE.

UF_UGMGR_ask_configured_rev (view source)

Defined in: uf_ugmgr.h

Overview
Finds the database tag of the configured revision of a specified part.

Environment
Internal and External

Required License(s)
gateway

```
int UF_UGMGR_ask_configured_rev
(
    UF_UGMGR_tag_t database_part_tag,
    UF_UGMGR_tag_t * part_revision
)
```

UF_UGMGR_tag_t	**database_part_tag**	Input	Database tag of the part whose configured revision we wish to find out.
UF_UGMGR_tag_t *	**part_revision**	Output	Database tag of the configured revision of the specified part.

UF_UGMGR_ask_dependent_files (view source)

Defined in: uf_ugmgr.h

Overview
Returns a list of the names of all dependent files and a count of the dependent files that are associated with the specified encoded part file name.

Environment
Internal and External

Required License(s)
gateway

```
int UF_UGMGR_ask_dependent_files
(
    char encoded_name [ MAX_FSPEC_BUFSIZE ] ,
    int* file_count,
    char* * * file_names
)
```

char	**encoded_name [MAX_FSPEC_BUFSIZE]**	Input	Encoded form of the part file name.

int*	**file_count**	Output	Number of dependent files associated with the specified part.
char* **	**file_names**	Output to UF_*free*	List of the names of all the dependent files associated with the specified part. The list should be freed after use by calling UF_free_string_array()

UF_UGMGR_ask_export_directory (view source)

Defined in: uf_ugmgr.h

Overview

Asks for the name of the export directory (if one exists) in which all dependent files associated with the specified part are placed.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_ask_export_directory
(
    tag_t part_tag,
    char export_dir_name [ MAX_FSPEC_BUFSIZE ]
)
```

tag_t	**part_tag**	Input	NX part tag.
char	**export_dir_name [MAX_FSPEC_BUFSIZE]**	Output	Export directory name. Declare this parameter with array size MAX_FSPEC_BUFSIZE.

UF_UGMGR_ask_family_member_handles (view source)

Defined in: uf_ugmgr.h

Overview

Returns the pre-allocated part and part rev handle values for the member. The API will return NULL values for the handles if the member id and the revision id do not match the stored values.

For the input member_id:
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x

Environment
Internal and External

History
Originally released in NX 3.0.2

Required License(s)
gateway

int UF_UGMGR_ask_family_member_handles
(
tag_t family_tag,
const char* member_id,
const char* member_rev_id,
char* * member_part_handle,
char* * member_partrev_handle
)

tag_t	**family_tag**	Input	family tag of the part family template
const char*	**member_id**	Input	member id, i.e. DB_PART_NO value of the family member
const char*	**member_rev_id**	Input	member rev id, DB_PART_REV value of the family member
char* *	**member_part_handle**	Output to UF_*free*	part handle of the family member This must be freed by calling UF_free()
char* *	**member_partrev_handle**	Output to UF_*free*	part rev handle of the family member This must be freed by calling UF_free()

UF_UGMGR_ask_file_export_status (view source)

Defined in: uf_ugmgr.h

Overview
Asks the status of export of associated files. If the status is TRUE, file export is enabled and associated files can be exported, else file export is disabled and no associated files can be exported.

Environment
Internal and External

History
Originally released in V17.0

Required License(s)
gateway

int UF_UGMGR_ask_file_export_status
(
logical* status
)

logical*	**status**	Output	Ask the current status of export of files. If the status is TRUE, file export is enabled, else file export is disabled.

UF_UGMGR_ask_folder_name (view source)

Defined in: uf_ugmgr.h

Overview
Finds the name of the folder corresponding to the specified database folder tag. You must specify the tag of a folder that already exists.

Environment
Internal and External

Required License(s)
gateway

```
int UF_UGMGR_ask_folder_name
(
    UF_UGMGR_tag_t folder,
    char folder_name [ UF_UGMGR_NAME_BUFSIZE ]
)
```

UF_UGMGR_tag_t	**folder**	Input	Database tag of the folder whose name we wish to know
char	**folder_name [UF_UGMGR_NAME_BUFSIZE]**	Output	Name corresponding to the specified folder tag. Declare this parameter with array size UF_UGMGR_NAME_BUFSIZE.

UF_UGMGR_ask_id_display_rule (view source)

Defined in: uf_ugmgr.h

Overview
Returns your current ID display rule. If none are defined for the current user or the last chosen ID display rule no longer exists in the PDM, then "$PDM default" is returned.

Environment
Internal and External

History
Originally released in NX 2.0.1

Required License(s)
gateway

```
int UF_UGMGR_ask_id_display_rule
(
    char * * id_display_rule
)
```

char * *	**id_display_rule**	Output to UF_*free*	Your current ID display rule. String must be freed after use with UF_free().

UF_UGMGR_ask_new_alternate_part_no (view source)

Defined in: uf_ugmgr.h

Overview
Returns a pointer to a registered program or NULL if none is registered.

Environment
Internal and External

History
Originally released in NX 2.0.1

Required License(s)
gateway

```
int UF_UGMGR_ask_new_alternate_part_no
(
    UF_UGMGR_new_alternate_part_no_fn_t * func
)
```

UF_UGMGR_new_alternate_part_no_fn_t *	**func**	Output	Pointer to registered function to invoke.

UF_UGMGR_ask_new_dataset_name (view source)

Defined in: uf_ugmgr.h

Overview
Returns the user's method for generating a new dataset name registered using UF_UGMGR_reg_new_dataset_name.
Finds the user's method for generating a new dataset name. The supplied callback is executed in response to the "Assign" button in the new part dialog in NX Manager. If a NULL is used then the program will be un-registered.
In the registered callback, a call to UF_UI_lock_ug_access is necessary prior to calling any interactive API functions. Also - it is not necessary to call UF_initialize in the callback. It will raise an error if called.

Note: This routine, if registered, will be called in preference to the existing Team Center Engineering user exit routine.

Environment
Internal and External

See Also
UF_UGMGR_new_dataset_name_t

History
This function was originally released in V15.0.

Required License(s)
gateway

int UF_UGMGR_ask_new_dataset_name
(
UF_UGMGR_new_dataset_name_fn_t * func
)

UF_UGMGR_new_dataset_name_fn_t *	**func**	Input	Pointer to registered function to invoke

---

UF_UGMGR_ask_new_part_rev (view source)

Defined in: uf_ugmgr.h

Overview
Finds the user's method for generating a new part revision name. The supplied callback is executed in response to the "Assign" button in the new part dialog in NX Manager. If a NULL is used then the program will be un-registered.
In the registered callback, a call to UF_UI_lock_ug_access is necessary prior to calling any interactive API functions. Also - it is not necessary to call UF_initialize in the callback. It will raise an error if called.

Note: This routine, if registered, will be called in preference to the existing Team Center Engineering user exit routine.

Environment
Internal and External

See Also
UF_UGMGR_reg_new_part_rev
UF_UGMGR_new_part_rev_t

History
This function was originally released in V15.0

Required License(s)
gateway

int UF_UGMGR_ask_new_part_rev
(
UF_UGMGR_new_part_rev_fn_t * func
)

UF_UGMGR_new_part_rev_fn_t *	**func**	Input	Pointer to registered function to invoke.

UF_UGMGR_ask_object_type (view source)

Defined in: uf_ugmgr.h

Overview

Queries the type of an object corresponding to the specified object database tag. You must specify the tag of a valid database object. Valid database objects are Folders, Parts and Part revisions. If the tag specifies an invalid object, then object_type has a value of UF_UGMGR_type_unknown.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_ask_object_type
(
    UF_UGMGR_tag_t object,
    UF_UGMGR_object_type_t* object_type
)
```

UF_UGMGR_tag_t	**object**	Input	Database tag of the object to query.
UF_UGMGR_object_type_t*	**object_type**	Output	Type of object corresponding to the specified tag. UF_UGMGR_type_folder UF_UGMGR_type_part UF_UGMGR_type_part_revision UF_UGMGR_type_unknown.

UF_UGMGR_ask_part_name_desc (view source)

Defined in: uf_ugmgr.h

Overview

Finds the part name and part description of a part in the Team Center Engineering database corresponding to the specified database part tag.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_ask_part_name_desc
(
    UF_UGMGR_tag_t database_part_tag,
    char part_name [ UF_UGMGR_NAME_BUFSIZE ] ,
    char part_desc [ UF_UGMGR_DESC_BUFSIZE ]
)
```

UF_UGMGR_tag_t	**database_part_tag**	Input	Database tag of the part whose name and description we require.
char	**part_name [UF_UGMGR_NAME_BUFSIZE]**	Output	Part name. Declare this parameter with array size UF_UGMGR_NAME_BUFSIZE.
char	**part_desc [UF_UGMGR_DESC_BUFSIZE]**	Output	Part description. Declare this parameter with array size UF_UGMGR_DESC_BUFSIZE.

UF_UGMGR_ask_part_number (view source)

Defined in: uf_ugmgr.h

Overview

Finds the part number corresponding to the specified database part tag. You must specify the tag of a part that already exists in the Team Center Engineering database.

For the output part_number:
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_ask_part_number
(
    UF_UGMGR_tag_t part,
    char part_number [ UF_UGMGR_NAME_BUFSIZE ]
)
```

UF_UGMGR_tag_t	**part**	Input	Database tag of the part to query.
char	**part_number [UF_UGMGR_NAME_BUFSIZE]**	Output	Part number corresponding to the specified database part tag. Declare this parameter with array size UF_UGMGR_NAME_BUFSIZE.

UF_UGMGR_ask_part_revision_id (view source)

Defined in: uf_ugmgr.h

Overview

Finds the name of the part revision (i.e.. the part revision id) corresponding to the specified database tag of the part revision.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_ask_part_revision_id
(
    UF_UGMGR_tag_t part_revision,
    char revision_id [ UF_UGMGR_NAME_BUFSIZE ]
)
```

UF_UGMGR_tag_t	**part_revision**	Input	Database tag of the part revision.
char	**revision_id [UF_UGMGR_NAME_BUFSIZE]**	Output	Name of the part revision corresponding to the specified part revision tag. Declare this parameter with array size UF_UGMGR_NAME_BUFSIZE.

UF_UGMGR_ask_part_tag (view source)

Defined in: uf_ugmgr.h

Overview

Finds the tag of a part in the database corresponding to the specified part number.
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x

If the function is successful, the output argument contains the database tag of the part. If the function is unsuccessful, then the output argument is UF_UGMGR_null_tag.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_ask_part_tag
(
    char * part_number,
    UF_UGMGR_tag_t * database_part_tag
)
```

char *	**part_number**	Input	Part number.
UF_UGMGR_tag_t *	**database_part_tag**	Output	Database tag of the part corresponding to the specified part number.

UF_UGMGR_ask_partrev_part_tag (view source)

Defined in: uf_ugmgr.h

Overview

Finds the tag of a part in the database corresponding to the specified part revision tag. If the function is successful, the output argument contains the database tag of the part. If the function is unsuccessful, then the output argument is UF_UGMGR_null_tag.

Environment

Internal and External

History

Released in NX3.0

Required License(s)

gateway

```
int UF_UGMGR_ask_partrev_part_tag
(
    UF_UGMGR_tag_t database_part_rev_tag,
    UF_UGMGR_tag_t * database_part_tag
)
```

UF_UGMGR_tag_t	**database_part_rev_tag**	Input	Database tag of the part revision
UF_UGMGR_tag_t *	**database_part_tag**	Output	Database tag of the part corresponding to the specified part revision tag.

UF_UGMGR_ask_product_assemblies (view source)

Defined in: uf_ugmgr.h

Overview

Returns an array of all Product Assemblies currently recognised by the session. A Product Assembly is any Partrev whose corresponding Team Center Engineering Item Revision contains one or more associated Variant Rules.

The Product Assemblies are returned in CLI format (e.g. "@DB/partname/A")

Environment

Internal and External

History

Originally released in V19.0

Required License(s)
gateway

int UF_UGMGR_ask_product_assemblies
(
int * n_prod_assys,
char * * * products
)

int *	**n_prod_assys**	Output	Number of product assemblies the session knows about
char * * *	**products**	Output to UF_*free*	Current set of product assemblies, in CLI form. Free with UF_free_string_array

---

UF_UGMGR_ask_root_folder (view source)

Defined in: uf_ugmgr.h

Overview
Returns the database tag of the root folder for the current user. This database tag is returned in the argument folder_tag. If an error occurs, NULL is returned for the tag.

Environment
Internal and External

Required License(s)
gateway

int UF_UGMGR_ask_root_folder
(
UF_UGMGR_tag_t * folder_tag
)

UF_UGMGR_tag_t *	**folder_tag**	Output	The database tag of the root folder for the current user.

UF_UGMGR_ask_saveas_dataset_info (view source)

Defined in: uf_ugmgr.h

Overview
Returns the user's method that was registered using UF_UGMGR_reg_saveas_dataset_info that is used to generate the non-master dataset naming information during "Save As New Item" or "Save As New Item Revision" operation on the master part.

Environment

Internal and External

History
Originally released in NX 2.0.5

Required License(s)
gateway

int UF_UGMGR_ask_saveas_dataset_info
(
UF_UGMGR_saveas_dataset_info_fn_t * func
)

UF_UGMGR_saveas_dataset_info_fn_t *	**func**	Output	Pointer to the user supplied registered Open C API program

---

UF_UGMGR_ask_user_folder (view source)

Defined in: uf_ugmgr.h

Overview
Returns the database tag of the root folder for the specified user. This database tag is returned in the argument folder_tag. If an error occurs, NULL is returned for the tag.

Environment
Internal and External

History
First released in V19.0

Required License(s)
gateway

int UF_UGMGR_ask_user_folder
(
const char * user_name,
UF_UGMGR_tag_t * folder_tag
)

const char *	**user_name**	Input	the name of the user for whom the root folder tag is required
UF_UGMGR_tag_t *	**folder_tag**	Output	The database tag of the root folder for the specified user

---

UF_UGMGR_ask_user_role (view source)

Defined in: uf_ugmgr.h

Overview

Returns the current NX Manager user role to be used in filtering associated files when they are exported from the Team Center Engineering database.
For example, if Manufacturing has been specified as the user role, only manufacturing-related files are exported from Team Center Engineering when you open a part file from NX.
For more information, see the NX Manager documentation.

This should not be confused with the Team Center Engineering role, which is a separate concept.

For more information on the meaning and behaviour of the NX Manager role, see the NX Manager online help documentation.

Environment
Internal and External

Required License(s)
gateway

```
int UF_UGMGR_ask_user_role
(
    char role [ UF_UGMGR_ROLE_BUFSIZE ]
)
```

char	**role [UF_UGMGR_ROLE_BUFSIZE]**	Output	User role. Declare this parameter with array size UF_UGMGR_ROLE_BUFSIZE.

UF_UGMGR_ask_variant_configurations_for_display (view source)

Defined in: uf_ugmgr.h

Overview
This routine returns the current Variant Display Options for a particular displayed part.

The first argument of the function is the displayed part for which you want the Variant Display Options.

An array of Variant Configurations is returned - each of these contains a Product Assembly and a Variant Rule.

A parallel array of logicals indicates whether each of these Variant Configurations is active.

The occurrence tree of this displayed part is configured by the active Variant Configurations - so some part occurrences may be suppressed. Note that this set of Variant Display Options (like the part occurrence tree) is specific to a particular displayed part - so a Variant Configuration may be active for one displayed part and not for another, and an instance whose part occurrence is suppressed in this displayed part may not be suppressed in another displayed part.

Environment
Internal and External

History
Originally released in V19.0

Required License(s)
gateway

int UF_UGMGR_ask_variant_configurations_for_display
(
tag_t available_displayed_part,
int * n_variants,
UF_UGMGR_variant_configuration_t * * variants,
logical * * selected
)

tag_t	**available_displayed_part**	Input	Part that returned information is about.
int *	**n_variants**	Output	Number of variant configurations available.
UF_UGMGR_variant_configuration_t * *	**variants**	Output to UF_*free*	Available variant configurations. Free with UF_free.
logical * *	**selected**	Output to UF_*free*	Parallel array of logicals: true if corresponding variant configuration is to be selected for this part, false otherwise. Free with UF_free.

UF_UGMGR_ask_variant_configurations_for_load (view source)

Defined in: uf_ugmgr.h

Overview
Returns the current Variant Load Options.

An array of Variant Configurations is returned - each of these contains a Product Assembly and a Variant Rule.

A parallel array of logicals indicates whether each of these Variant Configurations is active.

When an assembly is loaded into the session, all these active Variant Configurations will be applied to it.

Environment
Internal and External

History
Originally released in V19.0

Required License(s)
gateway

int UF_UGMGR_ask_variant_configurations_for_load
(

```
    int * n_variants,
    UF_UGMGR_variant_configuration_t * * variants,
    logical * * selected
)
```

int *	**n_variants**	Output	Number of variant configurations available.
UF_UGMGR_variant_configuration_t * *	**variants**	Output to UF_*free*	n_variants Available variant configurations. Free with UF_free.
logical * *	**selected**	Output to UF_*free*	n_variants Parallel array of logicals: true if corresponding variant configuration is to be selected for load, false otherwise. Free with UF_free.

UF_UGMGR_assign_alternate_part_id (view source)

Defined in: uf_ugmgr.h

Overview

This routine generates a new alternate part number by calling a TCEng user exit. Depending on the implemented user exit, the name returned may be dependent on part number, part type, context or ID type. A logical is returned alongside the new alternate item ID or multifield key and alternate revision ID names. This logical indicates whether the caller is allowed to change the suggested alternate item and revision IDs after they have been generated without calling this function. The behavior of the default user exit is to return true at all times.

For the output alt_item_id:
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x

Environment

Internal and External

History

Originally released in NX 2.0.1

Required License(s)

gateway

```
int UF_UGMGR_assign_alternate_part_id
(
    const tag_t part_tag,
    const char * context,
    const char * id_type,
    char * * alt_item_id,
    char * * alt_rev_id,
    logical * modifiable
)
```

const tag_t	**part_tag**	Input	Tag of part to assign alternate ID.
const char *	**context**	Input	Context for which an alternate number is being requested.
const char *	**id_type**	Input	ID type of part for which number is being requested. May be NULL.
char * *	**alt_item_id**	Output to UF_*free*	Alternate Item ID. The string must freed after use with UF_free().
char * *	**alt_rev_id**	Output to UF_*free*	Alternate Revision ID. The string must freed after use with UF_free().
logical *	**modifiable**	Output	Boolean; is the alternate ID allowed to be modified?

UF_UGMGR_assign_copy_dset_name (view source)

Defined in: uf_ugmgr.h

Overview
Generates the dataset name for a copied non-master part based on a call to the Team Center Engineering user exit, USER_copied_dataset_name.

Environment
Internal and External

See Also
UF_UGMGR_assign_new_dset_name.

History
This function was originally released in V15.0

Required License(s)
gateway

```
int UF_UGMGR_assign_copy_dset_name
(
    const char old_owner [ UF_UGMGR_PARTNO_BUFSIZE ] ,
    const char old_owner_revision [ UF_UGMGR_PARTREV_BUFSIZE ] ,
    const char * dataset,
    const char * app_type,
    const char * rel_type,
    const char new_owner [ UF_UGMGR_PARTNO_BUFSIZE ] ,
    const char new_owner_revision [ UF_UGMGR_PARTREV_BUFSIZE ] ,
    char model_name [ UF_UGMGR_NAME_BUFSIZE ] ,
    logical * modifiable
)
```

const char	**old_owner [UF_UGMGR_PARTNO_BUFSIZE]**	Input	String name of old owning part
const char	**old_owner_revision [UF_UGMGR_PARTREV_BUFSIZE]**	Input	String name of revision of old owning part

const char *	**dataset**	Input	String name of old dataset
const char *	**app_type**	Input	String name of application type
const char *	**rel_type**	Input	String name of relation type
const char	**new_owner [UF_UGMGR_PARTNO_BUFSIZE]**	Input	String name of new owning part or empty string
const char	**new_owner_revision [UF_UGMGR_PARTREV_BUFSIZE]**	Input	String name of revision of new owning part or empty string
char	**model_name [UF_UGMGR_NAME_BUFSIZE]**	Output	String name of new dataset
logical *	**modifiable**	Output	Whether result is user modifiable

---

UF_UGMGR_assign_new_dset_name (view source)

Defined in: uf_ugmgr.h

Overview
Generates the dataset name for a new non-master part based on a call to the Team Center Engineering user exit, USER_new_dataset_name.

Environment
Internal and External

See Also
UF_UGMGR_assign_copy_dset_name

Required License(s)
gateway

```
int UF_UGMGR_assign_new_dset_name
(
    const char owner [ UF_UGMGR_PARTNO_BUFSIZE ] ,
    const char owner_revision [ UF_UGMGR_PARTREV_BUFSIZE ] ,
    const char * app_type,
    const char * rel_type,
    const char * basis_string,
    char model_name [ UF_UGMGR_NAME_BUFSIZE ] ,
    logical * modifiable
)
```

const char	**owner [UF_UGMGR_PARTNO_BUFSIZE]**	Input	String name of owning part
const char	**owner_revision [UF_UGMGR_PARTREV_BUFSIZE]**	Input	String name of revision of owning part

const char *	**app_type**	Input	String name of application type
const char *	**rel_type**	Input	String name of relation type
const char *	**basis_string**	Input	String name of default name (may be an empty string)
char	**model_name [UF_UGMGR_NAME_BUFSIZE]**	Output	String name of new dataset
logical *	**modifiable**	Output	Whether result is user modifiable

---

UF_UGMGR_assign_part_number (view source)

Defined in: uf_ugmgr.h

Overview

This routine generates a new part number by calling an Team Center Engineering user exit, USER_new_item_id(). Depending on the implementation of the user exit, the new part number may be based on an existing part. In which case the number of that part must be supplied as the basis_part_num parameter. The type of the part may also be required. The behavior of the default user exit is to ignore the part type.
A logical is returned alongside the new part number. This logical indicates whether the caller is allowed to change the part number after it has been generated without calling this function. The behavior of the default user exit is to return true at all times.

For the basis and new part_num:
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x

Environment

Internal and External

See Also

UF_UGMGR_validate_part_rev

Required License(s)

gateway

```
int UF_UGMGR_assign_part_number
(
    const char basis_part_num [ UF_UGMGR_PARTNO_BUFSIZE ] ,
    const char * part_type,
    char part_num [ UF_UGMGR_PARTNO_BUFSIZE ] ,
    logical * modifiable
)
```

const char	**basis_part_num [UF_UGMGR_PARTNO_BUFSIZE]**	Input	Number of a previous part that the new number may be based on.

const char *	**part_type**	Input	Type of part for which number is being requested. May be NULL.
char	**part_num [UF_UGMGR_PARTNO_BUFSIZE]**	Output	Part number.
logical *	**modifiable**	Output	Boolean; is part_num allowed to be modified by caller.

UF_UGMGR_assign_part_rev (view source)

Defined in: uf_ugmgr.h

Overview

This routine generates a new part revision identifier by calling an Team Center Engineering user exit, USER_new_rev_id(). The part number for which a new revision is required must be supplied. The type of the part may also be supplied if required by the implementation of the user exit. The new revision identifier is returned together with a logical indicating whether the caller is allowed to change it without calling this function.

For the input part_num for which revision is requested:
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x

Environment

Internal and External

See Also

UF_UGMGR_validate_part_rev

Required License(s)

gateway

```
int UF_UGMGR_assign_part_rev
(
    const char part_num [ UF_UGMGR_PARTNO_BUFSIZE ] ,
    const char * part_type,
    char part_rev [ UF_UGMGR_PARTREV_BUFSIZE ] ,
    logical * modifiable
)
```

const char	**part_num [UF_UGMGR_PARTNO_BUFSIZE]**	Input	Part number for which revision is requested.
const char *	**part_type**	Input	Type of part for which new revision is requested. May be NULL.

char	**part_rev [UF_UGMGR_PARTREV_BUFSIZE]**	Output	Part revision. Declare this parameter with array size UF_UGMR_PARTREV_SIZE + 1
logical *	**modifiable**	Output	Boolean; is part_rev allowed to be modified by caller.

UF_UGMGR_attach_alternate (view source)

Defined in: uf_ugmgr.h

Overview

Attaches an alternate ID to the given part. The alternate is defined given by the input parameters context, id_type, alt_item_id, alt_rev_id, alt_name and alt_desc and attached to the part with tag part_tag.

For the input alt_item_id:
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x

Environment

Internal and External

History

Originally released in NX 2.0.1

Required License(s)

gateway

```
int UF_UGMGR_attach_alternate
(
    const tag_t part_tag,
    const char * context,
    const char * id_type,
    const char * alt_item_id,
    const char * alt_rev_id,
    const char * alt_name,
    const char * alt_desc,
    const logical is_default
)
```

const tag_t	**part_tag**	Input	Tag of part to attach alternate to.
const char *	**context**	Input	Context of the alternate to attach.
const char *	**id_type**	Input	ID type of the alternate to attach.
const char *	**alt_item_id**	Input	Alternate Item ID of the alternate to attach.
const char *	**alt_rev_id**	Input	Alternate Revision ID of alternate to attach.

const char *	**alt_name**	Input	Alternate Name of alternate to attach. May be NULL.
const char *	**alt_desc**	Input	Alternate Description of alternate to attach. May be NULL.
const logical	**is_default**	Input	True if this alternate ID should become the default ID, false otherwise.

UF_UGMGR_convert_file_name_to_cli (view source)

Defined in: uf_ugmgr.h

Overview

Converts a part name from the internal format to the command line input format. This routine is a null operation if not running NX Manager.

In case of Default Domain:
Master Part
From: "%UGMGR=V3.2 PH=itemUid PRH= itemRevUid PN= item_id PRN=item_revision_id RT="has shape" AT="UG master part file""
To: "@DB/<item_id>/<item_revision_id>"

Non-master Part
From: "%UGMGR=V3.2 PH=itemUid PRH= itemRevUid PN= item_id PRN=item_revision_id AN="dataset name" RT="relation type" AT="Dataset type" AUID=appuid"
To: "<@DB><separator><item_id><separator><item_revision_id><separator><Item Revision to dataset relation><dataset_name>"

e.g.,
From: "%UGMGR=3.2 PN=Peters-part -PRN=A..."
From: "@DB/peters-part/A/spec/sheet1"

In case of non-Default Domain:
Master Part
From: "%UGMGR=V3.2 PH=itemUid PRH= itemRevUid PN= MFK ID PRN=item_revision_id RT="has shape" AT="UG master part file""
To: "@DB/<MFK ID>/<item_revision_id>"

Non-master Part
From: "%UGMGR=V3.2 PH=itemUid PRH= itemRevUid PN= MFK ID PRN=item_revision_id AN="dataset name" RT="relation type" AT="Dataset type" AUID=appuid"
To: "<@DB><separator><MFK ID><separator><item_revision_id><separator><Item Revision to dataset relation><dataset_name>"

e.g.,
From: "%UGMGR=V3.2 PH=iyDp59Kwx6sBFD PRH=iyOp59Kwx6sBFD PN=",=item_id=test123, object_type=testItemType" PRN=A AN=test-alt1 RT="has altrep" AT="UG alternative rep" AUID=C3Jp59Kwx6sBFD"
To: "@DB/,=item_id=test123,object_type=testItemType/A"

Environment

Internal and External

History

This function was originally released in NX 5.0 and is mandatory if Longer IDs functionality is enabled NX/Manager

Required License(s)

gateway

int UF_UGMGR_convert_file_name_to_cli
(
const char * internal_name,
char * * cli_name
)

const char *	**internal_name**	Input	Internal format "%UGMGR=3.2 PN=Peters-part -PRN=A..."
char * *	**cli_name**	Output to UF_*free*	Command Line Input format, e.g., "@DB/peters-part/A/spec/sheet1"

UF_UGMGR_convert_name_from_cli (view source)

Defined in: uf_ugmgr.h

Overview

Converts a part file name from the command line input format to the internal form. This routine is a null operation if not running NX Manager and does not give an error.

In case of Default Domain:
Master Part
From: "@DB/<item_id>/<item_revision_id>"
To: "%UGMGR=V3.2 PH=itemUid PRH= itemRevUid PN= item_id PRN=item_revision_id RT="has shape" AT="UG master part file""

Non-master Part
From: "<@DB><separator><item_id><separator><item_revision_id><separator><Item Revision to dataset relation><dataset_name>"
To: "%UGMGR=V3.2 PH=itemUid PRH= itemRevUid PN= item_id PRN=item_revision_id AN="dataset name" RT="relation type" AT="Dataset type" AUID=appuid"

e.g.,
From: "@DB/peters-part/A/spec/sheet1"
To: "%UGMGR=3.2 PN=Peters-part -PRN=A..."

In case of non-Default Domain:
Master Part
From: "@DB/<MFK ID>/<item_revision_id>"
To: "%UGMGR=V3.2 PH=itemUid PRH= itemRevUid PN= MFK ID PRN=item_revision_id RT="has shape" AT="UG master part file""

Non-master Part
From: "<@DB><separator><MFK ID><separator><item_revision_id><separator><Item Revision to dataset relation><dataset_name>"
To: "%UGMGR=V3.2 PH=itemUid PRH= itemRevUid PN= MFK ID PRN=item_revision_id AN="dataset name" RT="relation type" AT="Dataset type" AUID=appuid"

e.g.,
From: "@DB/,=item_id=test123,object_type=testItemType/A"
To: "%UGMGR=V3.2 PH=iyDp59Kwx6sBFD PRH=iyOp59Kwx6sBFD PN=",=item_id=test123, object_type=testItemType" PRN=A AN=test-alt1 RT="has altrep" AT="UG alternative rep" AUID=C3Jp59Kwx6sBFD"

Note that UF_PART_new, UF_PART_open and UF_PART_save_as accept an input part file name in the command line input format without conversion.

Environment
Internal and External

See Also
UF_UGMGR_convert_file_name_to_cli.

Required License(s)
gateway

```
int UF_UGMGR_convert_name_from_cli
(
    const char * cli_name,
    char * * internal_name
)
```

const char *	**cli_name**	Input	Command Line Input format, e.g., "@DB/peters-part/A/spec/sheet1" Note: The separator used must not be a character which is already used in the part name, part revision, part file type or part file name. Valid separators are /@#$^&()|[]{}.
char * *	**internal_name**	Output to UF_*free*	Internal format "%UGMGR=3.2 PN=Peters-part -PRN=A...". The caller must free this string by calling UF_free

---

UF_UGMGR_create_component_part (view source)

Defined in: uf_ugmgr.h

Overview
Extended function for 'UF_ASSEM_create_component_part' with additional parameter for Part Type.
Create a new part with the given part type, moves selected objects to it, then adds an instance of it to the parent part.
Any other transferrable objects upon which the given objects depend are also moved into the component.

Environment
Internal and External

History
Originally released in NX 3.0.4

Required License(s)
gateway

```
int UF_UGMGR_create_component_part
(
    tag_t parent_part,
    const char * new_part_name,
```

```
    const char * refset_name,
    const char * instance_name,
    int units,
    int layer,
    double origin [ 3 ] ,
    double csys_matrix [ 6 ] ,
    int n_objects,
    tag_t * objects,
    const char* part_type,
    tag_t * instance
)
```

tag_t	**parent_part**	Input	Tag of parent part
const char *	**new_part_name**	Input	Name of new component part
const char *	**refset_name**	Input	Name of reference set
const char *	**instance_name**	Input	Name of instance to add to parent part
int	**units**	Input	1 = MM, 2 = Inches
int	**layer**	Input	-1 = original 0 = use work layer 1-255 = use specified layer
double	**origin [3]**	Input	Position in parent part where the instance is to be created.
double	**csys_matrix [6]**	Input	Orientation of the instance
int	**n_objects**	Input	Number of objects in the "objects" array.
tag_t *	**objects**	Input	Pointer to an array of tags of objects that should be moved to the new component part.
const char*	**part_type**	Input	Part type for the new component Part
tag_t *	**instance**	Output	Tag of instance

UF_UGMGR_decode_part_file_name (view source)

Defined in: uf_ugmgr.h

Overview

Decodes the specified NX Manager part file name into its constituent elements - the part number, part revision, part file type, and part file name. If a particular element is not present in the encoded input string, the corresponding output argument is the empty string "".
The encoded_name input argument can be any one of the following:

An internal form string such as that returned from the function UF_UGMGR_encode_part_filename or other functions that return part filenames (e.g. UF_PART_ask_part_name).

A command line input form string such as that returned from UF_UGMGR_convert_file_name_to_cli.

A string input by a user of an Open API program.

For the output part_number:
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x

Environment
Internal and External

See Also
UF_UGMGR_encode_part_filename
UF_UGMGR_convert_file_name_to_cli

History
This function was originally released in NX 5.0.and is mandatory if Longer IDs functionality is enabled NX/Manager

Required License(s)
gateway

```
int UF_UGMGR_decode_part_file_name
(
    const char * encoded_name,
    char part_number [ UF_UGMGR_PARTNO_BUFSIZE ] ,
    char part_revision [ UF_UGMGR_PARTREV_BUFSIZE ] ,
    char part_file_type [ UF_UGMGR_FTYPE_BUFSIZE ] ,
    char part_file_name [ UF_UGMGR_FNAME_BUFSIZE ]
)
```

const char *	**encoded_name**	Input	Encoded form of the part file name. Declare this parameter with array size MAX_FSPEC_BUFSIZE.
char	**part_number [UF_UGMGR_PARTNO_BUFSIZE]**	Output	Part number. Declare this parameter with array size UF_UGMGR_PARTNO_BUFSIZE.
char	**part_revision [UF_UGMGR_PARTREV_BUFSIZE]**	Output	Part revision. Declare this parameter with array size UF_UGMGR_PARTREV_BUFSIZE.
char	**part_file_type [UF_UGMGR_FTYPE_BUFSIZE]**	Output	Part file type. Declare this parameter with array size UF_UGMGR_FTYPE_BUFSIZE.
char	**part_file_name [UF_UGMGR_FNAME_BUFSIZE]**	Output	Part file name. Declare this parameter with array size UF_UGMGR_FNAME_BUFSIZE.

UF_UGMGR_encode_part_filename (view source)

Defined in: uf_ugmgr.h

Overview

Encodes the input arguments into an NX Manager part file name. The resultant part file name can then be used in any other Open API subroutine that accepts a part file name (e.g., UF_PART_open, UF_PART_new). The following rules apply to the input arguments:

If you specify a value of NULL, the empty string "", or "master" for the part_file_type argument, the function assumes that you wish to encode the part as a master part file. In this case, the part_file_name field should be specified as NULL or the empty string, "".

If you specify a part file type of "specification" or "manifestation", you must also specify a name for the specification or manifestation part file.

The total combined length for the part_number, part_revision, and part_file_name fields should not exceed 50 characters. In addition, the maximum permissible length for the part_number is 26 characters, 16 characters for the part_revision, and 32 characters for the part_file_name.

For the input part_number:
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x
And the encoded part filename would be containing the MFK.

This function should not be called for CPD shapeDesign parttype

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_encode_part_filename
(
    const char * part_number,
    const char * part_revision,
    const char * part_file_type,
    const char * part_file_name,
    char encoded_name [ MAX_FSPEC_BUFSIZE ]
)
```

const char *	**part_number**	Input	Part number.
const char *	**part_revision**	Input	Part revision.
const char *	**part_file_type**	Input	Part file type. Should be NULL, the empty string "", "master", "specification" or "manifestation". Note that "master", "specification", and "manifestation" must be spelt using lower-case letters only.

const char *	**part_file_name**	Input	Part file name. Only required if the part_file_type is "specification" or "manifestation".
char	**encoded_name [MAX_FSPEC_BUFSIZE]**	Output	Encoded form of part file name. Declare this parameter with array size MAX_FSPEC_BUFSIZE.

UF_UGMGR_find_configured_rev (view source)

Defined in: uf_ugmgr.h

Overview

Finds the database tag of the child part revision in an Assembly using the immediate parent revision tag and child part tag as the input. If the parent assembly is precise, returns the precise configured item revision tag of child part. If the parentrev_tag is NULL_TAG the child_part_rev will be the revision configured as per the Revision rule.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_find_configured_rev
(
    UF_UGMGR_tag_t parentrev_tag,
    UF_UGMGR_tag_t childpart_tag,
    UF_UGMGR_tag_t * child_part_rev
)
```

UF_UGMGR_tag_t	**parentrev_tag**	Input	Database tag of the precise parent revision whose child part's revision we wish to find out. If passed a NULL_TAG, the child revsion returned will be based on revision rule
UF_UGMGR_tag_t	**childpart_tag**	Input	Database tag of the child part whose revision we wish to find out.
UF_UGMGR_tag_t *	**child_part_rev**	Output	Database tag of the configured revision of the child part in an Assembly.

UF_UGMGR_find_product_assemblies (view source)

Defined in: uf_ugmgr.h

Overview

Searches for all Product Assemblies that contain any of the parts listed in the "displayed parts" argument as components or components of

subassemblies. It adds these to the list of Product Assemblies that the session knows about. Note that the Product Assemblies are not actually returned.

If 0 is used as the first argument, the routine will search for all Product Assemblies that contain any of the current session's current list of displayed parts (this is likely to contain all loaded parts that have ever been displayed parts in the session).

Environment
Internal and External

History
Originally released in V19.0

Required License(s)
gateway

```
int UF_UGMGR_find_product_assemblies
(
    int n_displayed_parts,
    const tag_t displayed_parts [ ]
)
```

int	**n_displayed_parts**	Input	Number of displayed parts to search from. If n_displayed_parts is 0, all current displayed parts will be searched and the displayed_parts argument ignored.
const tag_t	**displayed_parts [ ]**	Input	n_displayed_parts Displayed parts to search from.

UF_UGMGR_generate_base_file_name (view source)

Defined in: uf_ugmgr.h

Overview
This routine returns a base file name.

Environment
Internal and External

History
Originally released in NX 10.0

Required License(s)
gateway

```
int UF_UGMGR_generate_base_file_name
(
    const char * partSpec,
    char * * baseFileName
)
```

const char *	**partSpec**	Input	part specification
char * *	**baseFileName**	Output	Base file name

UF_UGMGR_get_creation_parameters (view source)

Defined in: uf_ugmgr.h

Overview

This routine returns an array of creation parameters which are needed to create a part of the given part type in Teamcenter. Each parameters is represented by a UF_UGMGR_ATTR_info_t struct. The parameter can be a create descriptor, or a key field, or both. The part_type cannot be NULL or Empty String.

Environment

Internal and External

History

Originally released in NX 10.0

Required License(s)

gateway

```
int UF_UGMGR_get_creation_parameters
(
    const char * part_type,
    int * num_info,
    UF_UGMGR_ATTR_info_t * * info
)
```

const char *	**part_type**	Input	Type of part.
int *	**num_info**	Output	Number of creation parameters.
UF_UGMGR_ATTR_info_t * *	**info**	Output to UF_*free*	Array of the creation parameters. The array must be freed after use with UF_free().

UF_UGMGR_initialize (view source)

Defined in: uf_ugmgr.h

Overview

Initializes the NX Manager Open API environment (in addition to initializing the standard NX Open API environment) and allocates an Open API execute license. If you do not call UF_UGMGR_initialize, subsequent calls to Open API fail with an error code of UF_err_program_not_initialized. This function must be the first function that you call after you have declared your variables.

The arguments argc and argv are for external Open API programs only. If you have an internal Open program, you may pass in 0 for argc, and NULL for argv.

Environment
Internal and External

Required License(s)
gateway

```
int UF_UGMGR_initialize
(
    int argc,
    const char * * argv
)
```

int	**argc**	Input	Argument count. This will typically be the argc parameter passed to your main program from the system.
const char * *	**argv**	Input	Pointers to argument character strings. This will typically be the argv parameter passed to your main program from the system.The strings must always be in the user's locale. The input data should not be UTF8 data.

UF_UGMGR_invoke_pdm_server (view source)

Defined in: uf_ugmgr.h

Overview
Passes program control to Team Center Engineering, thereby allowing you to make Team Center Engineering ITK calls. When the Open API UF_UGMGR_invoke_pdm_server is called, it makes a call to the function USER_invoke_pdm_server the Team Center Engineering user exit shared library. Note the difference between these function names.

In your ITK program, the USER_invoke_pdm_server() function should contain all the ITK functions to implement the required functionality. The function should be linked into the shared library libuser_exits.sl. Refer to the Information Manager Integration Tool Kit Reference Manual for additional information about implementing these ITK functions using User Exits.

The required prototype for the function USER_invoke_pdm_server() is:

extern void USER_invoke_pdm_server(int input_code,
char input_string,
int output_code,
char output_string);

The USER_invoke_pdm_server() ITK function has the same arguments as the UF_UGMGR_invoke_pdm_server Open API function.

The output_string argument must be allocated in the ITK function USER_invoke_pdm_server() using the C programming language malloc() function, and freed in the UF_UGMGR_invoke_pdm_server Open API code by calling UF_free().

The input_code argument to USER_invoke_pdm_server can be used as a switch,

so that you may call a different ITK function depending upon the value of input_code. When the USER_invoke_pdm_server function terminates, program control returns back from your ITK program to your Open API program. You may specify two arguments to return to your Open API program - an integer output_code and a string output_string.

Suppose that you have na Open API program from which you wish to call an Team Center Engineering ITK program. The ITK program takes a part number as input, and returns the date that the part was created or the date it was modified, depending upon the value of the input_code argument to the function USER_invoke_pdm_server(). This could be implemented as follows:

The Open API program asks the user for a part number, and then calls the Open API UF_UGMGR_invoke_pdm_server(), that passes control to the ITK function USER_invoke_pdm_server(). The values returned from this ITK function are printed.

Environment
Internal and External

Required License(s)
gateway

```
int UF_UGMGR_invoke_pdm_server
(
    int input_code,
    char * input_string,
    int * output_code,
    char * * output_string
)
```

int	**input_code**	Input	User defined input code passed to your Team Center Engineering ITK function USER_invoke_pdm_server().
char *	**input_string**	Input	User defined input string passed to your Team Center Engineering ITK function USER_invoke_pdm_server().
int *	**output_code**	Output	User defined return code that is returned from the Team Center Engineering ITK USER_invoke_pdm_server() function on completion of ITK calls.
char * *	**output_string**	Output to UF_*free*	User defined string that is returned from the Team Center Engineering ITK USER_invoke_pdm_server() function on completion of ITK calls. This output string will be allocated by your ITK program, and must be freed by your Open API program by calling UF_free().

UF_UGMGR_list_config_rules (view source)

Defined in: uf_ugmgr.h

Overview
Returns the total number of configuration rules available in your system, and a list of the names of all these configuration rules.

Environment
Internal and External

Required License(s)
gateway

int UF_UGMGR_list_config_rules
(
int * count,
char * * * config_rules
)

int *	**count**	Output	Number of configuration rules.
char * * *	**config_rules**	Output to UF_*free*	Array of the names of all the configuration rules. The array must be freed after use by calling UF_free_string_array().

UF_UGMGR_list_contexts (view source)

Defined in: uf_ugmgr.h

Overview
This routine returns an array of names of all the contexts available for the given part type in the TCEng database. part_type cannot be NULL or Empty String

Environment
Internal and External

History
Originally released in NX 2.0.1

Required License(s)
gateway

int UF_UGMGR_list_contexts
(
const char * part_type,
int * count,
char * * * contexts
)

const char *	**part_type**	Input	Type of part.
int *	**count**	Output	Number of contexts in the database.
char * * *	**contexts**	Output to UF_*free*	Array of names of all the contexts. The array must be freed after use with UF_free().

UF_UGMGR_list_folder_contents (view source)

Defined in: uf_ugmgr.h

Overview
This function only returns tags of folders and parts contained in the input folder. You must specify the database tag of a folder that already exists.

Environment
Internal and External

Required License(s)
gateway

```
int UF_UGMGR_list_folder_contents
(
    UF_UGMGR_tag_t folder,
    int * count,
    UF_UGMGR_tag_t * * folder_contents
)
```

UF_UGMGR_tag_t	**folder**	Input	Database tag of the folder whose contents you wish to be listed.
int *	**count**	Output	The number of objects in the specified folder.
UF_UGMGR_tag_t * *	**folder_contents**	Output to UF_*free*	An array of database tags of the folder and part objects in the specified folder. This array must be freed after use using UF_free().

UF_UGMGR_list_id_display_rules (view source)

Defined in: uf_ugmgr.h

Overview
Returns the total number of ID display rules available to the user and a list of the names of all these ID display rules.

Environment
Internal and External

History
Originally released in NX 2.0.1

Required License(s)
gateway

```
int UF_UGMGR_list_id_display_rules
(
    int * count,
    char * * * id_display_rules
```

)

int *	**count**	Output	Number of ID display rules.
char * * *	**id_display_rules**	Output to UF_*free*	Array of the names of the ID display rules available to the current user. The array must be freed after use by calling UF_free_string_array().

UF_UGMGR_list_id_types (view source)

Defined in: uf_ugmgr.h

Overview
This routine returns an array of names of all the ID types for a given context in the TCEng database. If context is a NULL string, the list returned covers all available id_types. part_type cannot be NULL or empty string

Environment
Internal and External

History
Originally released in NX 2.0.1

Required License(s)
gateway

```
int UF_UGMGR_list_id_types
(
    const char * part_type,
    const char * context,
    int * count,
    char * * * id_types
)
```

const char *	**part_type**	Input	Type of part.
const char *	**context**	Input	Context for which the list of id_types is being requested. May be NULL.
int *	**count**	Output	Number of ID types for this context.
char * * *	**id_types**	Output to UF_*free*	Array of the names of the ID types which are valid for the given context. The array must be freed after use with UF_free().

UF_UGMGR_list_part_rev_files (view source)

Defined in: uf_ugmgr.h

Overview

Finds the number, list of names, and list of file types of non-master part files that a specified part revision contains. The possible file types are "specification", "manifestation" and "Foreign_Datasets". Foreign_Datasets can only be used if TC preference "TC_NX_Foreign_Datasets" is defined for "mult-CAD" type.

For example, a part revision may have two non-master files of type "specification" named "drawing001" and "drawing002", and one file of type "manifestation" called "NCtoolpath001". The output arguments of the function store the following information:

file_count: 3
file_types: specification specification manifestation
file_names: drawing001 drawing002 NCtoolpath001

Elements of the file_types and file_names arrays correspond (e.g., the third element of the file_types array is "manifestation-->that corresponds to the third element of the file_names array "NCtoolpath001").

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_list_part_rev_files
(
    UF_UGMGR_tag_t part_revision,
    int * file_count,
    char * * * file_types,
    char * * * file_names
)
```

UF_UGMGR_tag_t	**part_revision**	Input	Part revision.
int *	**file_count**	Output	Number of files.
char * * *	**file_types**	Output to UF_*free*	Array of file types. The array must be freed after use by calling UF_free_string_array().
char * * *	**file_names**	Output to UF_*free*	Array of file names. The array must be freed after use by calling UF_free_string_array().

UF_UGMGR_list_part_revisions (view source)

Defined in: uf_ugmgr.h

Overview

Asks for the number of revisions of a specified part and obtains a list of database tags of all of the part revisions.

Environment

Internal and External

Required License(s)
gateway

int UF_UGMGR_list_part_revisions
(
UF_UGMGR_tag_t part,
int * revision_count,
UF_UGMGR_tag_t * * revisions
)

UF_UGMGR_tag_t	**part**	Input	Database tag of the part to query.
int *	**revision_count**	Output	Number of revisions of the specified part.
UF_UGMGR_tag_t * *	**revisions**	Output to UF_*free*	Array of database tags of all the revisions of the specified part. The array must be freed after use by calling UF_free().

UF_UGMGR_list_parts_in_folder (view source)

Defined in: uf_ugmgr.h

Overview
This function returns tags of parts contained in the given folder. You must specify the database tag of a folder that already exists.
Note: This function will not return tags of part revisions.

Environment
Internal and External

History
Released in NX3.0

Required License(s)
gateway

int UF_UGMGR_list_parts_in_folder
(
UF_UGMGR_tag_t folder,
int * count,
UF_UGMGR_tag_t * * parts
)

UF_UGMGR_tag_t	**folder**	Input	Database tag of the folder whose part you wish to be listed.
int *	**count**	Output	The number of parts in the specified folder.

UF_UGMGR_tag_t * *	**parts**	Output to UF_*free*	An array of database tags of type UF_UGMGR_type_part or UF_UGMGR_type_part_revision in the specified folder. This array must be freed after use by calling UF_free().

UF_UGMGR_new_part_from_template (view source)

Defined in: uf_ugmgr.h

Overview

Creates a new part in the database using a template part as the basis for the part. If NULL is passed as the template name then the seed part specified in the defaults file is used. This function also requires a part type to be specified. If this is NULL then the part type specified in the defaults file is used. If no default part type is specified, then a type of 'Item' is used.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_new_part_from_template
(
    char encoded_part_name [ MAX_FSPEC_BUFSIZE ] ,
    char* part_type,
    char encoded_template_name [ MAX_FSPEC_BUFSIZE ] ,
    tag_t * part_tag
)
```

char	**encoded_part_name [MAX_FSPEC_BUFSIZE]**	Input	Encoded form of the part file name. Declare this parameter with array size MAX_FSPEC_BUFSIZE.
char*	**part_type**	Input	Part type
char	**encoded_template_name [MAX_FSPEC_BUFSIZE]**	Input	Encoded form of the template name. Declare this parameter with array size MAX_FSPEC_BUFSIZE.
tag_t *	**part_tag**	Output	NX tag of the newly created part.

UF_UGMGR_partrev_where_used (view source)

Defined in: uf_ugmgr.h

Overview
Performs a single-level where-used operation on the given part revision.
The current configuration rule will be used.

Environment
Internal and External

See Also

History
This function was originally released into V18.0

Required License(s)
gateway

int UF_UGMGR_partrev_where_used
(
UF_UGMGR_tag_t part_revision,
int * parent_revisions_count,
UF_UGMGR_tag_t * * parent_revisions
)

UF_UGMGR_tag_t	**part_revision**	Input	Database tag of the part_revision for which to perform a where-used.
int *	**parent_revisions_count**	Output	Number of revisions of the specified part.
UF_UGMGR_tag_t * *	**parent_revisions**	Output to UF_*free*	Array of database tags of all part revisions which use the specified part_revision according to the current configuration rule. The array must be freed after use by calling UF_free().

UF_UGMGR_refresh_assy_pdi_date (view source)

Defined in: uf_ugmgr.h

Overview
Not a published API as of now.

Environment
Internal

History
Originally released in NX4.0.1

Required License(s)
gateway

int UF_UGMGR_refresh_assy_pdi_date
(

```
    tag_t part_tag,
    logical traverse
)
```

tag_t	**part_tag**	Input
logical	**traverse**	Input

UF_UGMGR_reg_new_alternate_part_no (view source)

Defined in: uf_ugmgr.h

Overview
Stores a pointer for a user supplied registered callback. May be NULL to un-register a program.

Environment
Internal and External

History
Originally released in NX 2.0.1

Required License(s)
gateway

```
int UF_UGMGR_reg_new_alternate_part_no
(
    UF_UGMGR_new_alternate_part_no_fn_t func
)
```

UF_UGMGR_new_alternate_part_no_fn_t	**func**	Input	Pointer to the user supplied registered Open API callback.

UF_UGMGR_reg_new_dataset_name (view source)

Defined in: uf_ugmgr.h

Overview
Registers the user's method for generating a new dataset name. The supplied callback is executed in response to the "Assign" button in the New part or Save As dialog when new non-master or CAE part is being created.
If a NULL is used then the program will be un-registered.
In the registered callback, a call to UF_UI_lock_ug_access is necessary prior to calling any interactive API functions. Also - it is not necessary to call UF_initialize in the callback. It will raise an error if called.

Note: This routine, if registered, will be called in preference to the existing Teamcenter user exit routine.

Environment

Internal and External

See Also
UF_UGMGR_ask_new_dataset_name
UF_UGMGR_new_dataset_name_t

History
This function was originally released in V15.0.

Required License(s)
gateway

int UF_UGMGR_reg_new_dataset_name
(
UF_UGMGR_new_dataset_name_fn_t new_dataset_name_fn
)

UF_UGMGR_new_dataset_name_fn_t	**new_dataset_name_fn**	Input	Pointer to the user supplied registered Open API callback.

UF_UGMGR_reg_new_part_rev (view source)

Defined in: uf_ugmgr.h

Overview
Registers the user's method for generating a new part revision. The supplied callback is executed in response to the "Assign" button in the new part dialog in NX Manager. If a NULL is used then the program will be un-registered. In the registered callback, a call to UF_UI_lock_ug_access is necessary prior to calling any interactive API functions. Also - it is not necessary to call UF_initialize in the callback. It will raise an error if called.

Note: This routine, if registered, will be called in preference to the existing Team Center Engineering user exit routine.

Environment
Internal and External

See Also
UF_UGMGR_ask_new_part_rev
UF_UGMGR_new_part_rev_t

History
This function was originally released in V15.0.

Required License(s)
gateway

int UF_UGMGR_reg_new_part_rev
(
UF_UGMGR_new_part_rev_fn_t new_part_rev_fn
)

UF_UGMGR_new_part_rev_fn_t	**new_part_rev_fn**	Input	Pointer to the user supplied registered Open API program

---

UF_UGMGR_reg_saveas_dataset_info (view source)

Defined in: uf_ugmgr.h

Overview

Registers the user's method that is used to generate the non-master dataset naming information during "Save As New Item" or "Save As New Item Revision" operation on the master part.

If a NULL is used then the program will be un-registered.
Memory for the return array elements in the structure UF_UGMGR_saveas_dataset_info_s, namely dataset_name_list, copy_state, name_locked, validation_required, original_owner, and each character array in the dataset_name_list array must be allocated using UF_allocate_memory
in this method implementation.

Environment

Internal and External

History

Originally released in NX 2.0.5

Required License(s)

gateway

int UF_UGMGR_reg_saveas_dataset_info
(
UF_UGMGR_saveas_dataset_info_fn_t saveas_dataset_info_fn
)

UF_UGMGR_saveas_dataset_info_fn_t	**saveas_dataset_info_fn**	Input	Pointer to the user supplied registered Open C API program

---

UF_UGMGR_remove_product_assembly_part (view source)

Defined in: uf_ugmgr.h

Overview

Removes a Product Assembly from the list of those recognised by the current NX Manager session.

Specify the Product Assembly in CLI format.

Environment

Internal and External

History

Originally released in NX 5.0 and is mandatory if Longer IDs functionality is enabled NX/Manager

Required License(s)

gateway

int UF_UGMGR_remove_product_assembly_part
(
const char * product
)

const char *	**product**	Input	Name of the product assembly in CLI form.

UF_UGMGR_resolve_part_file_infos (view source)

Defined in: uf_ugmgr.h

Overview

This is the bulk replacement for all UF functions that deal with NX part file names and TC Item/ItemRev/Dataset,such as UF_UGMGR_encode_part_filename, UF_UGMGR_decode_part_file_name, UF_UGMGR_ask_dependent_files, UF_UGMGR_ask_part_tag, UF_UGMGR_list_part_revisions, UF_UGMGR_ask_part_number, UF_UGMGR_ask_part_revision_id, UF_UGMGR_list_part_rev_files, etc.

Environment

Internal and External

History

Originally released in NX 10.0

Required License(s)

gateway

int UF_UGMGR_resolve_part_file_infos
(
unsigned int partFileCount,
const UF_UGMGR_part_file_info_t* partFileInput,
UF_UGMGR_part_file_object_t* * * partFileObject
)

unsigned int	**partFileCount**	Input	Number of UF_UGMGR_part_file_info_t's to be processeed
const UF_UGMGR_part_file_info_t*	**partFileInput**	Input	UF_UGMGR_part_file_info_t's to be resolved
UF_UGMGR_part_file_object_t* * *	**partFileObject**	Output to UF_*free*	An array of pointers to UF_UGMGR_part_file_object_t. Each UF_UGMGR_part_file_object_s object returned corresponds to a UF_UGMGR_part_file_info_s(NX PartFile) that was resolved This array must be freed after use using UF_free()

UF_UGMGR_save_precise_assembly (view source)

Defined in: uf_ugmgr.h

Overview
To save the given assembly part as Precise Structure in the database
It will attempt to save the assembly and any of its sub-assembly structure
that are out-of-sync with its corresponding Precise BVR in the database.

Environment
Internal and External

History
Originally released in NX 5.0

Required License(s)
gateway

```
int UF_UGMGR_save_precise_assembly
(
    tag_t work_part_tag,
    logical traverse_children
)
```

tag_t	**work_part_tag**	Input	Part tag for the assembly that has to be saved
logical	**traverse_children**	Input	true if the sub-assemblies and nested sub-assemblies are considered for Save operation

UF_UGMGR_set_autolock_status (view source)

Defined in: uf_ugmgr.h

Overview
Allows setting of the autolock status for checking out of parts when modified.
This is the equivelent of the UGII_UGMGR_NOAUTOLOCK environment variable.
If the autolock status is true, a part will be automatically checked out
when it is loaded if it needs to be modified. Setting this value to false
will prevent the checkout from automatically happening.

Environment
Internal and External

See Also
UF_UGMGR_ask_autolock_status

History
This function was originally released in V16.0.2

Required License(s)
gateway

int UF_UGMGR_set_autolock_status
(
logical new_value
)

logical	**new_value**	Input	New value of the autolock status TRUE = part is automatically checked out on update. FALSE = part is not automatically checked out.

UF_UGMGR_set_clone_auto_trans (view source)

Defined in: uf_ugmgr.h

Overview
Registers an autotranslate function for use during Clone Based Import/Export. If the value of function is NULL, the default autotranslate function is registered.

Environment
Internal and External

See Also
UF_UGMGR_clone_auto_trans_f_t

History
This function was originally released in V15.0.

Required License(s)
gateway

int UF_UGMGR_set_clone_auto_trans
(
UF_UGMGR_clone_auto_trans_f_t
)

UF_UGMGR_clone_auto_trans_f_t	Input	Auto Translate Function

UF_UGMGR_set_config_rule (view source)

Defined in: uf_ugmgr.h

Overview
Sets the current configuration rule to be the specified rule. The specified configuration rule must be valid. A list of valid configuration rules can be obtained by calling the Open API UF_UGMGR_list_config_rules.

Environment
Internal and External

See Also
UF_UGMGR_list_config_rules

Required License(s)
gateway

int UF_UGMGR_set_config_rule
(
char* config_rule
)

char*	**config_rule**	Input	Configuration rule.

UF_UGMGR_set_default_folder (view source)

Defined in: uf_ugmgr.h

Overview
Sets the default folder into which parts are placed, given the database tag of the folder. You must specify the database tag of a folder that already exists. This function does not create the folder if you specify the tag of a non-existent folder.

Environment
Internal and External

Required License(s)
gateway

int UF_UGMGR_set_default_folder
(
UF_UGMGR_tag_t folder
)

UF_UGMGR_tag_t	**folder**	Input	Database tag of the folder.

UF_UGMGR_set_dialog_display (view source)

Defined in: uf_ugmgr.h

Overview
To set a static to allow sub-dialogs to be shown in Interactive User Function and GRIP only.

Environment
Internal

Required License(s)
gateway

int UF_UGMGR_set_dialog_display
(
logical display
)

logical	**display**	Input	TRUE to allow dialogs to display. False if dialogs are not to display.

UF_UGMGR_set_file_export_status (view source)

Defined in: uf_ugmgr.h

Overview
Turns the export of associated files on or off. Associated files are exported when the status argument to the function is true. If status is set to false, no associated files are exported.

Environment
Internal and External

Required License(s)
gateway

int UF_UGMGR_set_file_export_status
(
logical status
)

logical	**status**	Input	Turns the export of files on or off. If set to TRUE, file export is enabled; if set to FALSE, file export is disabled.

UF_UGMGR_set_id_display_rule (view source)

Defined in: uf_ugmgr.h

Overview
Sets the current ID display rule to be the given rule. The rule specified must be valid. A list of valid ID display rules for the current user may be obtained by calling the Open API UF_UGMGR_list_id_display_rules.

Environment
Internal and External

History
Originally released in NX 2.0.1

Required License(s)
gateway

```
int UF_UGMGR_set_id_display_rule
(
    const char * id_display_rule
)
```

const char *	**id_display_rule**	Input	ID display rule.

UF_UGMGR_set_user_role (view source)

Defined in: uf_ugmgr.h

Overview

Sets the NX Manager user role to be used in filtering associated files when they are exported from the Team Center Engineering database. For example, if you specify Manufacturing as the user role, only manufacturing-related files are exported from Team Center Engineering when you open a part file from NX.

This should not be confused with the Team Center Engineering role, which is a separate concept.

For more information on the meaning and behaviour of the NX Manager role, see the NX Manager online help documentation.

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_set_user_role
(
    char * role
)
```

char *	**role**	Input	User role.

UF_UGMGR_set_variant_configurations_for_display (view source)

Defined in: uf_ugmgr.h

Overview

This routine sets the current Variant Display Options for a particular displayed part.

The first argument of the function is the displayed part for which you want to set the Variant Display Options.

You must supply an array of Variant Configurations - each of these contains a Product Assembly and a Variant Rule.

You must also supply a parallel array of logicals indicating whether each of these Variant Configurations is active for this displayed part.

The occurrence tree of this displayed part is configured by the active Variant Configurations - so some part occurrences may be suppressed. Note that this set of Variant Display Options (like the part occurrence tree) is specific to a particular displayed part - so a Variant Configuration may be active for one displayed part and not for another, and an instance whose part occurrence is suppressed in this displayed part may not be suppressed in another displayed part.

It is generally envisaged that one would call UF_UGMGR_ask_variant_configurations_for_display then change the array "selected" slightly and then call UF_UGMGR_set_variant_configurations_for_display to apply the changes.

Environment
Internal and External

History
Originally released in V19.0

Required License(s)
gateway

```
int UF_UGMGR_set_variant_configurations_for_display
(
    tag_t available_displayed_part,
    int n_variants,
    const UF_UGMGR_variant_configuration_t * variants,
    logical selected [ ]
)
```

tag_t	**available_displayed_part**	Input	Part for which to set variant configurations.
int	**n_variants**	Input	Number of variant configurations available.
const UF_UGMGR_variant_configuration_t *	**variants**	Input	Available variant configurations.
logical	**selected [ ]**	Input	Parallel array of logicals: true if corresponding variant configuration is selected for this part, false otherwise.

UF_UGMGR_set_variant_configurations_for_load (view source)

Defined in: uf_ugmgr.h

Overview
This routine sets the current Variant Load Options.

You must provide an array of Variant Configurations (preceded by a

separate argument giving its length). Each Variant Configuration contains a Product Assembly and a Variant Rule.

You must also provide a parallel array of logicals indicating whether each of these Variant Configurations is selected for load.

It is generally envisaged that one would call
UF_UGMGR_ask_variant_configurations_for_load
then change the array "selected" slightly and then call
UF_UGMGR_set_variant_configurations_for_load
to apply the changes.

Environment
Internal and External

History
Originally released in V19.0

Required License(s)
gateway

```
int UF_UGMGR_set_variant_configurations_for_load
(
    int n_variants,
    const UF_UGMGR_variant_configuration_t variants [ ] ,
    logical selected [ ]
)
```

int	**n_variants**	Input	Number of variant configurations available.
const UF_UGMGR_variant_configuration_t	**variants [ ]**	Input	Available variant configurations.
logical	**selected [ ]**	Input	Parallel array of logicals: true if corresponding variant configuration is selected for load, false otherwise.

UF_UGMGR_terminate (view source)

Defined in: uf_ugmgr.h

Overview
Obsolete from v12.0: just call UF_terminate

Required License(s)
gateway

```
int UF_UGMGR_terminate
(
    void
)
```

UF_UGMGR_validate_alternate_part_id (view source)

Defined in: uf_ugmgr.h

Overview

Validates the alternate ID selected for a new alternate of the current part.
Both the alternate item ID or multifield key and alternate revision ID names are validated. This function may reject the supplied identifier, accept it or amend it. The action is reflected in the value of status. This function calls a TCEng user exit function to perform the validation. The user exit defines what constitutes valid alternate item and revision names for an alternate ID and may depend on the part number, part type, context and ID type.

The behavior of the default stub supplied with TCEng as part of the user exit library is to always returns a status of USER_valid_name, with the output arguments modified_item_id and modified_rev_id set to empty strings. In effect, the default behaviour represents unconditional acceptance of the user input alternate item and revision names.

For the input alt_item_id and output modified_item_id:
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x

Environment

Internal and External

History

Originally released in NX 2.0.1

Required License(s)

gateway

```
int UF_UGMGR_validate_alternate_part_id
(
    const tag_t part_tag,
    const char * alt_item_id,
    const char * alt_rev_id,
    const char * context,
    const char * id_type,
    char * * modified_item_id,
    char * * modified_rev_id,
    UF_UGMGR_alt_id_status_t * status,
    char * * reason
)
```

const tag_t	**part_tag**	Input	Tag of part to assign the alternate ID to.
const char *	**alt_item_id**	Input	Alternate Item ID. May be NULL.
const char *	**alt_rev_id**	Input	Alternate Revision ID. May be NULL.
const char *	**context**	Input	Context for which alternate name is being validated. May be NULL.

const char *	**id_type**	Input	ID type for which alternate name is being validated. May be NULL.
char * *	**modified_item_id**	Output to UF_*free*	Modified Alternate Item ID. Will be an empty string if no amendment. The string must be freed after use with UF_free.
char * *	**modified_rev_id**	Output to UF_*free*	Modified Alternate Revision ID. Will be an empty string if no amendment. The string must be freed after use with UF_free.
UF_UGMGR_alt_id_status_t *	**status**	Output	Enum code indicating: UF_UGMGR_alt_id_valid - accept UF_UGMGR_alt_id_invalid - reject UF_UGMGR_alt_id_modified - input modified.
char * *	**reason**	Output to UF_*free*	Reason. The string must be freed after use with UF_free.

UF_UGMGR_validate_part_rev (view source)

Defined in: uf_ugmgr.h

Overview

Validates the part number and revision identifiers selected for a new part revision. This function may reject the supplied identifiers, accept them or amend them. The action is reflected in the value of status. This function calls the Team Center Engineering user exit function USER_validate_item_rev_id() to perform the validation.

The user exit defines what constitutes a valid combination of number and revision and depending on its implementation may take the type of the part into account.

The behavior of the default stub supplied with Team Center Engineering as part of the user exit library always returns a status of USER_valid_id, with the output arguments modified_item_id and modified_rev_id set to an empty string. In effect this represents unconditional acceptance of the user input.

For the input and modified part_num:
In case of Default Domain: it is Team Center Engineering item ID.
In case of non-Default Domain: it is the multifield key.
e.g. ,=item_id=001, object_type=Document
,=item_id=001, object_type=SupplierPart, supplier_code=x

Environment

Internal and External

Required License(s)

gateway

```
int UF_UGMGR_validate_part_rev
(
    const char part_num [ UF_UGMGR_PARTNO_BUFSIZE ] ,
    const char part_rev [ UF_UGMGR_PARTREV_BUFSIZE ] ,
    const char * part_type,
    char modified_part_num [ UF_UGMGR_PARTNO_BUFSIZE ] ,
    char modified_part_rev [ UF_UGMGR_PARTREV_BUFSIZE ] ,
    UF_UGMGR_partno_status_t * status
)
```

const char	**part_num [UF_UGMGR_PARTNO_BUFSIZE]**	Input	Part number. Declare this parameter with array size UF_UGMR_PARTNO_SIZE+1.
const char	**part_rev [UF_UGMGR_PARTREV_BUFSIZE]**	Input	Part revision. Declare this parameter with array size UF_UGMR_PARTREV_SIZE+1.
const char *	**part_type**	Input	Type of part for which number and revision is being validated.
char	**modified_part_num [UF_UGMGR_PARTNO_BUFSIZE]**	Output	Modified part number. Not set if no amendment.
char	**modified_part_rev [UF_UGMGR_PARTREV_BUFSIZE]**	Output	Modified part revision. Not set if no amendment.
UF_UGMGR_partno_status_t *	**status**	Output	Enum code indicating, UF_UGMGR_partno_valid - accept UF_UGMGR_partno_invalid - reject UF_UGMGR_partno_modified - input modified.

UF_UGMGR_validate_string (view source)

Defined in: uf_ugmgr.h

Overview

Checks if string is valid against the current Teamcenter Target Encoding.
The string will be considered valid if - when converted to the Target Encoding - its length does not exceed the given limit and all of its characters and can be represented in the Target encoding.

Environment

Internal and External

History

Originally released in NX 9.0

Required License(s)

gateway

```
int UF_UGMGR_validate_string
(
    const char* inputString,
    unsigned int lengthLimit,
    logical* hasInvalidCharacters
)
```

const char*	**inputString**	Input
unsigned int	**lengthLimit**	Input
logical*	**hasInvalidCharacters**	Output
