TECHNICAL DOCUMENT

Project No: ----

Company: ----

Project Title: IPS_DATA_UPLOAD TECHNICAL DOCUMENTATION

RELEASED

Document Info

Revision: 6.7.0

Issued Date: 12th January 2007

Reference: IPS_DATA_UPLOAD_6_7_0_TSD.doc

CIRCULATION

No.	***Copies to***	***Location***
1	Project Library	Internal Community Archive,

DOCUMENT HISTORY

Version	*Date*	*Author*	*Description*
6.5.0a	20/11/2006	Guy Bursell	Initial Technical Release for IPS_DATA_UPLOAD version 6.5.0. Technical Review and Training
6.5.0	24/11/2006	Guy Bursell	Updated from Review and Feedback.
6.6.0	29/11/2006	Guy Bursell	Added Checkin/Checkout functionality Added boms mode missing data field 'OccQty' to data fields table in document.
6.7.0	12/01/2007	Guy Bursell	Fixed error when check-in an object that is checked-out to another user. Enabled config/data fields to set ownership of a BV/BVR on creation. Fixed Signal 11 error when uploading large BOMs on HP/UX.

SIGNOFF

Name	*Company*	*Sign/Date*

REFERENCE DOCUMENTS

Document	*Revision*	*Date*	*Doc Reference*

CONTENTS

1. EXECUTIVE SUMMARY

The following document provides technical documentation for using the IPS_DATA_UPLOAD utility developed by UGS, UK. This utility is used to provide bulk data upload and data cleanup capability in Teamcenter Engineering.

The document outlines the functional modules in IPS_DATA_UPLOAD:

- Items
- Forms
- Datasets
- Folders
- BOMs
- Relations
- List of Values (LOVs)
- Identifiers
- Projects
- Manufacturing
- Classification

Each module has various configuration and data definition options, all of which are described in this document.

The IPS_DATA_UPLOAD utility is available on Windows 2000/XP/2003 platform as well as HP/UX 11x and Solaris UNIX platforms.

2. TECHNICAL OVERVIEW

The IPS_DATA_UPLOAD utility is a generic all-purpose tool used to import data into Teamcenter Engineering. It is capable of importing various types of objects and data and operates in a number of modes to achieve this:

MODE	DESCRIPTION
Items	Used to import Items, ItemRevisions and set their object properties. Also used to set Item Master and ItemRevision Master form attributes.
Forms	Used to import and populate Forms and attach them to Items or ItemRevision objects.
Datasets	Used to import dataset objects and their associated files. Datasets can be orphaned or attached to Item or ItemRevision objects.
Folders	Used to import folders. Folders can imported into structures.
BOMs	Used to import and build BOM structures and BOM related properties.
Relations	Used to allow imported data to be related or attached together using specific relationships defined in Teamcenter Engineering.
List of Values (LOVs)	Used to create and populate List of Values.
Identifiers	Used to set Alternate Identifiers on Items and ItemRevisions.
Projects	Used to create, update and assign Projects
Manufacturing	Used to Create and Update Teamcenter Manufacturing objects and to create and update Manufacturing BOM structures.
Classification	Create, Update and Delete Classification (In-Class) In-Class Objects (ICOs).

Table 2.1 Outline list of operation modes

The ability to use the `ips_data_upload` utility requires the following pre-requisites:

1. Knowledge of the Teamcenter Engineering data model Teamcenter Administration.
2. Operating System
3. Business Data being imported.

The `ips_data_upload` utility itself allows a number of ways of setting configuration and input data options; via command line arguments, a separate configuration file and even the ability to place configuration options in the input data file!

If command line arguments are supplied they will override the configuration file and the data file. The following order of precedence should be observed when specifying configuration options:

1. Command Line Argument
2. Data file configuration option
3. Configuration file configuration option
4. IMAN Preference Option.

2.1 USAGE

When running the utility, there are several command line arguments that it takes. These arguments take the format:

-option=<*value*>

The arguments are used to provide login user details, operating mode and data input file and configuration files as shown in the table below:

OPTION	DESCRIPTION
-h	Displays the full usage
-u=<***user_id***>	Where <***user_id***> is the iMAN login user ID
-p=<***password***>	Where <***password***> is the iMAN users' password
-g=<***group***>	Where <***group***> is the users' group to use
-m=<***mode***>	Specifies the mode of operation, such as **items**, **bom, datasets** or **relations**
-i=<***input_data_file***>	Where <***input_data_file***> is the data file to upload
-cfg=<***config_file***>	Where <***config_file***> is the configuration options file
-o=<***output_file***>	Where <***output_file***> is the output log file. If not supplied the default filename will be ***input_data_file.log***
-r=<***repeat_file***>	Where is the repeat file which will contain a copy of any data lines which caused an error. If not supplied the default filename will be ***input_data_file.rep***
-debug=MAX	Turns on program debug putput when running. Can be any of the following: 1|2|4|8|16|32 | MAX Typically this is set to **MAX**

Table 2.2 Command line options

2.2 INPUT DATA FILE

The input data file contains all of the information to be imported into Teamcenter Engineering. This file is a delimited file, typically using a [~] (tilde) character. The format of the input data file follows a number of rules:

1. Any line beginning with a [#] character or left blank is ignored.
2. The first line of the data input file is always a header line. This line is denoted by starting with an exclamation mark [!].
3. The delimiter used in the input data file is identified as the character immediately following the exclamation mark [!] on the header line.
4. The name of the data fields to use are specified immediately after the exclamation mark and delimiter on the header line [!~].
5. The header line must precede any data lines.

   An example header line is shown over page.

An example of a header line defined using the tilde [~] as the field separator is shown below.

```
!~ItemID~RevID~Desc~Name~Owner~Group
```

In this example, there are 6 fields defined using case sensitive keywords. An example of the Header line followed by a data input line is shown below:

```
!~ItemID~RevID~Desc~Name~Owner~Group
GUY12345~A~The Item Description~ITEM-NAME~mr_user~operations
```

The above data line with create or find the Item GUY12345, Revision A, and set its Item Description, Item Name and Owning User and Owning Group properties to the values supplied.

2.3 CONFIGURATION OPTIONS

Configuration options are generally defined in the configuration file, although some can be placed at the top of the data file before the header line. There are options to control a variety of functions and take the format:-

OPTION NAME = <*value*>

The <value> may be user values, such as an Item or Form type for instance, or predefined keywords such as ON or OFF for example, used for switching options. An example of defining configuration options is shown below.

```
CREATE ITEMS = ON
FORM TYPE    = Mfg Data Form
```

2.4 RUNNING AN IMPORT

In order to run a data import, a Teamcenter Engineering sourced environment is required. This involves setting a number of environment variables and is achieved as shown below, on the Windows platform. These same environment variables are also required on other platforms although the method of executing is slightly different.

```
C:\>SET IMAN_ROOT=C:\UGSAPPS\TCENG0913
C:\>SET IMAN_DATA=\\SERVER\SHARE\iman_data
C:\>%IMAN_DATA%\iman_profilevars
C:\>
```

3. TROUBLESHOOTING

A log file is always produced when IPS_DATA_UPLOAD is run. This log file will report all successes, failures and warnings. Where a failure or warning is encountered, information about this event will also be shown in the log file. The log file will have the file extension: `.log`

Where errors or failures are encountered, a repeat file is generated at the same time. This file will contain the line(s) from the data input file where an error or failure was encountered. The repeat file will have the file extension: `.rep`

In some remote cases, where an error being encountered is complex or not reported, it may be useful to have further detailed information. In this instance, a system log (.syslog) is generated and automatically stored, typically, in the `%TEMP%` folder location on Windows.

In order for the syslog file to provide meaningful information, it is important the accompanying `.pdb` file (Microsoft program debug database) exists in the same location as the IPS_DATA_UPLOAD executable. This is only required on the Windows platform.

Low level debugging information can also be generated when the utility is run providing a trace of the operation of the utility. The following environment variables should be set in the sourced Teamcenter Engineering environment the utility is running in and takes the format:

SET DEBUG_MAX=MAX

SET DEBUG_FILE_NAME=<*filename*>

```
C:\>SET DEBUG_MAX=MAX
C:\>SET DEBUG_FILE_NAME=c:\debug.log
C:\>
```

This will then create a debug trace in the above file which is typically very useful to UGS Technical personnel in resolving any problems.

4. ITEMS MODE

This section outlines the input data field definitions, configuration options and usage information for the items mode. The items mode is used to import data around the following criteria:

- Create Items, Item Revisions and set/update their properties
- Apply Release Status
- Populate Item and ItemRevision Master forms

This mode is entered by default if no mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=items
```

4.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to

`Items`

FIELD	DESCRIPTION
`ItemID`	Item ID
`Name`	Applies to Item if no Rev supplied, otherwise applies the Rev Name.
`Desc`	Applies to Item if no Rev supplied, otherwise applies the Rev Description.
`ItemName`	Specifically Item Name
`ItemDesc`	Specifically Item Description
`Uom`	Unit of Measure
`NewItemID`	New Item ID to change existing Item ID to or "save as" to depending on settings of CREATE ITEMS/UPDATE ITEMS
`ItemType`	Specifies the Item Type to use.

The following input data fields are valid in this mode and are specifically related to

`Revisions`

FIELD	DESCRIPTION
`RevID`	Revision ID
`RevName`	Specifically Rev Name
`RevDesc`	Specifically Rev Description
`NewRevID`	New Rev ID to change existing Rev ID to or "save as" to depending on settings of CREATE REVS/UPDATE REVS

The following input data fields are valid in this mode and are specifically related to

Owner Properties

FIELD	DESCRIPTION
CreateDate	Creation Date
ModifyDate	Last Modified Date
Owner	Owning User
Group	Owning Group
LastModBy	Last Modified by user
OwningSite	Owning Site. Use a value of "LOCAL" to unset the site.

The following input data fields are valid in this mode and are specifically related to

Miscellaneous

FIELD	DESCRIPTION
Folder	Folder Name to put new items in
Ignore	Used to Ignore a field

The following input data fields are valid in this mode and are specifically related to

Status

FIELD	DESCRIPTION
Status	Release status name
ReleaseDate	Release date (for both status object and workspace object).
EffStartDate	Effectivity Start Date
EffEndDate	Effectivity End Date
UnitStart	Unit Serial Number end
BOStatus	Unit Serial Number end
BOReleaseDate	Release status for Based On Item/Revision
EffEndItem	Effectivity End Item ID

The following input data fields are valid in this mode and are specifically related to

`Item and ItemRevision Master Forms`

FIELD	DESCRIPTION
`IM:<attr>`	To set an Item Master form attribute.
`IMD:<attr>`	To set an Item Master form date attribute.
`RM:<attr>`	To set an Item Revision Master form attribute
`RMD:<attr>`	To set an Item Revision Master form date attribute

NOTES: Item and Item Revision Master Forms

- The attribute name (replacing <attr> above) must be the real name from the database and not the display name.

- If the attribute is defined in the database as a date then use IMD: or RMD:. Then the date format configuration option (`DATE FORMAT`) can be used in the configuration file to define the format of the dates supplied in the data file.

The following input data fields are valid in this mode and are specifically related to

`Checkin/Checkout/Cancel Checkout`

FIELD	DESCRIPTION
`CheckOutID`	Checkout Change ID (MAX 32 chars)
`CheckOutReason`	Checkout Reason/Comment text (MAX 240 chars)

4.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to **Items/Revisions**

FIELD	DESCRIPTION
CREATE ITEMS	ON|OFF flag to control creation of Items
UPDATE ITEMS	ON|OFF flag to control updating Items
CREATE REVS	ON|OFF flag to control creation of Revisions
UPDATE REVS	ON|OFF flag to control updating Revisions
UPDATE LATEST REV	ON|OFF flag to use attach to latest rev/update the latest rev.
DEFAULT REV	Default rev, if not supplied, when creating a new Item
ITEM TYPE	Item Type to create new Items as
DEFAULT OWNER	Default Owner, if not supplied in the data input file
DEFAULT GROUP	Default Group, if not supplied in the data input file
UPDATE ITEM ATTACHMENTS	ON|OFF flag to control if Item attachments have their names updated to reflect a new ItemID. Default is ON
UPDATE REV ATTACHMENTS	ON|OFF flag to control if Revision attachments have their names updated to reflect a new ItemID and/or RevID. Default is ON
AUTO GENERATE ITEM ID	ON|OFF flag to control if ItemID should be auto generated if ON and ItemID field is empty. Requires an ItemID naming rule is in force in the Business Modeller.
DATE FORMAT	Describes the format date fields are in. For example: **%d-%b-%Y %H:%M** = 30-Apr-2003 07:50 **%d/%m/%Y** = 30/04/2003 **%y%m%d** = 03/04/30 Note: The format is the same as that used by the UNIX **date** command. All of the options can be found by using **man date** at the UNIX prompt. The default is format is: 30-Apr-2003 07:50 ***See Appendix A for all DATE FORMAT values available.***

The following configuration options are valid in this mode and are specifically related to **Miscellaneous**

FIELD	DESCRIPTION
MODE	Re-set the mode. **OBSOLETED**
FOLDER NAME	Default folder name for new items
LOG FILE	Override the default log file name
REPEAT FILE	Override the default repeat file name
FOLDER TYPE	Default Folder Type for folder creation/insertion
NEWLINE TOK	Newline token to allow multi line text entry into a Note form attribute type. Default if not specified is '%#10;'
SET BYPASS	ON|OFF flag to control if bypass privilege is available for DBA user. (General All-module option).
WILDCARD CHAR MULTI	Specifies an alternative character to use as a multiple wild character so that a '*' can be used in the item ID.
UPDATE MODIFY DATES	ON|OFF flag to control if the modification date on updated objects should be changed.
WILDCARD CHAR SINGLE	Specifies an alternate character to use as a single wild character

The following configuration options are valid in this mode and are specifically related to

Owning Site

FIELD	DESCRIPTION
DEFAULT OWNING SITE	Default Site name if not supplied in data file using OwningSite field.
UPDATE REMOTE OBJECTS	ON|OFF flag to control if objects which having Owning Site set should be updated

The following configuration options are valid in this mode and are specifically related to

Status

FIELD	DESCRIPTION
RELEASE STATUS	Name of default release status to use
RELEASE ITEM	ON|OFF flag to control if Item is to be released. (Default = OFF)
BASED ON STATUS	Default release status for Based On Item/Revision
RELEASE BASED ON ITEM	ON|OFF flag to control if Based On Item is to be released. (Default = OFF)
UPDATE REVS ATTACHS STATUS	ON|OFF flag to control if the Revisions' attachments are released, and release them if not. (Default = ON)
RELEASE RELATIONS	A Comma separated list of relations in which to release objects. The default is IMAN_specification.

RELEASE NAMED REFS	ON|OFF flag to control if forms stored as named references in a dataset are released when applying a status. (Default = ON).
UNRELEASE NAMED REFS	ON|OFF flag to control if forms stored as named references in a dataset are unreleased when unreleasing objects. (Default = ON).
UNRELEASE ITEMS	ON|OFF flag to control un-releasing Items
UNRELEASE REVS	ON|OFF flag to control un-releasing Revisions
UNRELEASE KEEP LAST	ON|OFF flag to control if the last status is to be kept, i.e. to clean up multiple statuses
CREATE STATUS	ON|OFF flag. Controls if non released objects should be released
UPDATE STATUS	Controls how released objects should have their status updated. Can be set to one of the following options:
OFF	Turns off ALL status updating
LAST	(Default). Only the last status object on a released object will be updated.
REPLACE ALL	Replaces all existing status objects on a released object with a new status with supplied values. Effectively performing an un-release and a re-release.
ADD	Adds a new status object to released objects. It will add the same status name more than one.
ADD UNIQUE	Like ADD, but only adds a new status if the object does not already have a status object with the supplied status name.
UNRELEASE	Un-releases objects. Note: to use this option, no Status field should be specified in the data fields.
BY NAME	Specifies that a specific status name only is to be updated. The specific status name is supplied using the configuration option UPDATE STATUS NAME
UDPATE STATUS NAME	Supplies a specific status name to match against and can be used in several of the UPDATE STATUS options as follows:
LAST	The last status will only be updated if it matches, otherwise any last status object will be updated.
ADD	Only adds a new status object if it matches, otherwise a new status will be added anyway.
ADD UNIQUE	Only adds a new status object if it matches, otherwise a new status will be added anyway or updated if already existing.
BY NAME	Specifies the actual status name which is to be modified.
USE PARENT STATUS	ON|OFF flag to control if each object being released should have a new status (OFF) or its parent's status attached (ON). (Default = ON)

RELEASE OBJECTS FILTER	Specifies a Class/Type filter to indicate which objects to release/update. The format is: **(Class)[!Type1!Type2]** for a class with excluded types or **Type1,Type2…** for a comma separated list of possible types to match
ITEM RELEASE RELATIONS	MUST BE SPECIFIED WHEN RELEASING VIA ITEMS TO RELEASE RELATIONS, Specifies a comma separated list of Revision relations to find objects in. The following are valid keywords:
REVISIONS	Specifies Revisions
BOMVIEW	Specifies BOMViews
ALL	Specifies all standard relations only, i.e. does not include Revisions and BOMViews. For example, to find BOMViews and Revisions use: BOMVIEW,REVISIONS
REV RELEASE RELATIONS	Specifies a comma separated list of Revision relations to find objects in. The following are valid keywords:
BOMVIEW	Specifies BOMViewRevisions
ALL	Specifies all standard relations only, i.e. does not include Revisions and BOMViewRevisions.
ACTIVITIES	Specifies any MEActivities associated with an MEOP Revision only.
DATASET RELEASE RELATIONS	Specifies a comma separated list of dataset relations to find objects in. The following keyword can be used:
ALL	Specifies all standard relations.
RELEASE DATASET RELATIONS	ON|OFF flag to control if dataset relations should be released/updated.
UNRELEASE DATASET RELATIONS	ON|OFF flag to control if dataset relations should be unreleased.
PURGE DATASET VERSIONS ON RELEASE	ON|OFF flag to control if the non-latest versions of datasets should be purged when applying a release status. The files in the Teamcenter volumes associated with the purged datasets are also removed.

CREATE CICO	ON|OFF flag to control if objects with NO current Checkout should have the Check Out set. Default = OFF.
UPDATE CICO OFF CHECK IN CHECK OUT CANCEL CHECK OUT	Options to define what action to take on an object which already has a Check-Out set. Valid options are: Turn off applying any update operation to an existing Check-Out. Perform a Check-In operation to an object with an existing Check-Out. Perform a Check-Out operation to an object. Perform a Cancel Checkout operation on a DATASET object only. Default = OFF.
CHECK OUT REASON	Reason/Comment text to add when performing a Check-Out on an object. DEFAULT = "Checked-Out by IPS_DATA_UPLOAD" if not specified and 'CheckOutReason' data field not used.
CICO OBJECTS FILTER	Specifies to comma separated list of object types valid to perform Check-In/Check-Out/Cancel Check-Out operation on. The format is: **Type1,Type2…** To exclude certain Classes/Types from the Check-In/Check-Out/Cancel Check-Out operations, the format is: **(Class)[!Type1!Type2]** If this Configuration option is not specified, all objects will be processed.

4.3 MODULE NOTES

The following points should be considered when using this mode:

1. In all modes of operation if the Item being operated on exists and has a different Type to that supplied as a configuration option or input data file field value the operation will not be performed.

2. If AUTO GENERATE ITEM ID = ON configuration option is set and the ItemID data field has been left blank, the following rules are followed in generating an automatic Item ID:
   - Naming Rule for item_id must be attached to the Item Type in Business Modeller.
   - Auto Generation must not be turned off in Teamcenter Engineering.
   - The top (first) pattern in item_id Naming Rule is always used.

3. To assign projects using the `items` mode, `UPDATE ITEMS` or `UPDATE REVS` must be set to `ON`. This is not required if using projects mode.

4. V9 projects can be created, updated and assigned using the `items` mode to get output from the `items` perspective. Alternatively, to get output from the project perspective use the `projects` mode.

4.3.1 RELEASE STATUS

If no release status date is given, the date is taken from the time of program run.

When setting a release status, all Specification attachments of a Item Revision, and any BOMViewRevisions (BVRs), are all given the same status as the rev. If multiple Revisions of the same item are to be given a status, they need to be listed in the data file in the order they are to be released as the dates are generated sequentially to avoid duplication.

If the release date is to be UPDATED to the current date/time then the header option is required but the column must be left empty.

If a dataset is being released, because it is in the Specification Relation of a Revision being released, and it has form objects attached as references then the forms will also get released.

If a Revision is already released then by default all of the Specification objects and BVRs will still be checked to see if they are released. If any attachments are found to not be released then the release status object attached to the Revision will be attached to these attachments. If just the release date is being updated this checking can be turned off by setting the configuration option `UPDATE REVS ATTACHS STATUS = OFF`.

It is also possible to:

- Control if non released objects should be released or if released objects should updated independently.
- Update status object properties without having to include the `Status` field in the data file.
- Update specific status objects by name.
- Add new status objects to already released objects to create multiple statuses.
- Release dataset relations.
- Specify an object Class and/or Type Filter to release/update or exclude specific objects.

4.3.2 DATE RESTRICTIONS

When using the following configuration option:

DATE FORMAT = %d/%m/%Y

The Operating System regional settings (on Windows this is called Locale) should be set to English UK or English US to maintain correct date translations. This is true even if using "`dd/mm/yyyy`". The results for given dates in none-English UK or English US region settings may translate as follows:

1/10/2004 -> 1/1/2004 and 1/12/2004 -> 1/1/2004

4.3.3 CHECK IN / CHECK OUT / CANCEL CHECK OUT

The following points should be noted when using the Check-In/Check-Out functionality:

- The Check-In/Check-Out user will be the username specified when running `ips_data_upload`.
- The following list is a valid list of classes of objects that can have a Check-Out/Check-In performed: `Item, ItemRevision, Form, Dataset, Folder, PSBOMView, PSBOMViewRevision`.
- This functionality is supported in modes: Items, bom, datasets, forms, folder_create, folder_insert, forms.
- When wanting to perform Check-in/Check-out operations on TcManufacturing objects (MEProcess+Rev, MEOP+Rev, MEWorkArea+Rev), specify these using ItemID and RevID data line fields. Use the `CREATE ITEMS = ON|OFF` and `CREATE REVS = ON|OFF` to control behaviour.
- The Check-in/Check-out functionality will check the corresponding mode UPDATE <OBJECT> configuration option to verify if the operation is to be performed. I.e.
  - Items; `UPDATE ITEMS = ON` must be set or no Checkin/Checkout will be performed.
  - The following additional configuration options are checked: `UPDATE REVS = ON, UPDATE DATASET = ON, UPDATE FORMS = ON, UPDATE BOMS = ON.`

4.3.4 UN-RELEASING ITEMS/REVISIONS

It is also possible to Un-Release Items, Revisions and attached BOMViews (BVs), BVRs and Specification objects. Any forms attached to datasets will also be un-released.

To instigate the Un-Release functionality simply specify a release status of `UNRELEASE`. To control whether items and/or revisions should be un-released there are configuration options `UNRELEASE ITEMS` and `UNRELEASE REVS` which can be set to `ON` or `OFF`. The option `UPDATE ITEMS` and `UPDATE REVS` must be on `ON` to un-release the corresponding objects.

It is possible to just keep the last release status if an object has multiple statuses by setting the configuration option `UNRELEASE KEEP LAST = ON`. The default is to remove all status objects.

It is also possible to un-release all the Revisions in an Item or just the latest Revision by just supplying the `ItemID`, so there is no need to specify any Revision IDs. If the configuration option `UPDATE LATEST REV = OFF` is set (the default) then all of the Items' Revisions will be un-released. If set to `ON` then just the latest revision will be un-released.

Once all of the status objects have been removed from the Item, Revisions and attachments an attempt will be made to delete the status objects themselves. If however any of the status objects are still attached to any other objects in the database an error `515110` will be written to the syslog file. This is not a problem since all of the required objects will have been un-released.

4.3.5 V7 EFFECTIVITIES

It is possible to set a single date range when using V7 effectivities.

The features of this are:

It automatically checks if V7 effectivity mode is active and sets the effectivity data accordingly, including pre V7 effectivity if using pre V7 mode.

It uses the existing field headers, i.e. EffStartDate and EffEndDate.

It will only set single dates in V7 effectivity mode.

It only sets dates in V7 effectivity mode and not units.

It will un-release objects released using V7 effectivities.

If V7 effectivity is being used and the EffStartDate and EffEndDate columns are included, but with no values then any existing pre V7 start and end dates will be applied as V7 effectivity dates, i.e. a pre V7 to V7 conversion.

It will update any existing effectivity dates, so long as there is only one set of dates. In V7 mode if there are more than one range of dates then they will not be updated and a message will be given stating "Cannot update multiple V7 effectivities".

4.3.6 CHANGING IDS

If the NewItemID data field is supplied and configuration options CREATE ITEMS = OFF is set but UPDATE ITEMS = ON is set, the Item ID will be changed to that supplied value.

If the NewRevID data field is supplied and configuration options CREATE REVS = OFF is set but UPDATE REVS = ON is set, the Revision ID will be changed to that supplied value.

When either the Item or Revision ID is updated all corresponding objects which by default have their names made up from a combination of the IDs will also be updated. This goes for Master forms and BOM Views for both the Item and the Revision and all Revisions in the item. These object names will only be updated if their old names matched the corresponding combination with the original Item and Rev IDs.

Other attachments will also have their names updated if they contain <ItemID><Separator><RevID> or just <ItemID> with the NewItemID and/or NewRevID. This is on by default, but can be disabled using the configuration options UPDATE ITEM ATTACHMENTS = OFF and UPDATE REV ATTACHMENTS = OFF.

For example:

If the Revision 000001/A is having the item ID changed to 999999 and Revision ID 1 then the dataset called TEXT-000001/A-001 would be renamed to TEXT-999999/1-001.

NOTE: The NewItemID and NewRevID fields control the updating of IDs and the revise/save as to new Revisions and Items depending on the setting of the relevant CREATE and UPDATE options. In short, assuming UPDATE is ON then if CREATE is ON new objects are created otherwise the ID on the existing objects are updated.

4.3.7 REVISING AND SAVE AS

If CREATE REVS =ON, the NewRevID data field is supplied and NewItemID is supplied (and is equal to the existing Item ID OR is empty), OR NewItemID has not been supplied, then the Revision is copied (Revised) within the Item. The new Revision ID is that of NewRevID or generated by Teamcenter Engineering if empty (using USER_new_revision_id()).

If the data field NewItemID is supplied and does NOT match the existing Item ID and CREATE ITEMS is ON, a new Item is created using that ID and the value of NewRevID if set. If NewItemID and/or NewRevID are empty, the default Teamcenter Engineering behaviour for ID generation is used (USER_new_id() and USER_new_revision_id()).

UPDATE ITEMS and/or UPDATE REVS set to ON will control the updating of the NEW Revision and OLD Item if a "revise" has occurred OR the NEW Item and NEW Revision if a "save as" occurred. Caution must be exercised when setting the create date of the new revision.

In Teamcenter Engineering V9+ a Deep Copy will also be performed on any new Items or Revisions to perform any Deep Copy Rules that may be set.

The data fields NewItemID and NewRevID control the updating of IDs and the Revise/Save As to new Revisions/Items depending on the setting of the relevant CREATE and UPDATE options. In short, assuming UPDATE is ON then if CREATE is ON new objects are created otherwise the ID on the existing objects are updated.

4.4 COMMAND LINE ARGUMENTS

The following command line arguments are valid in this mode and are specifically related to **Items**

FIELD	DESCRIPTION
-t=<*Item Type*>	Item Type TCENG PREFERENCE equivalent: **IPS_IMP_Item_Type**
-d=<*Default Rev*>	Specify Default Revision ID to use TCENG PREFERENCE equivalent: **IPS_IMP_Default_Rev**
-df=<*Date Format*>	Date format to use. TCENG PREFERENCE equivalent: **IPS_IMP_Date_Format** Configuration Option equivalent: **DATE FORMAT**
-folder=<*Folder Name*>	Target Folder Name for imported data. TCENG PREFERENCE equivalent: **IPS_IMP_Folder_Name**
-update_latest	No value required. Specify to Update the Latest Item Revision

4.5 EXAMPLES

Examples for items mode can be found in Appendix B.

5. BOMS MODE

This section outlines the input data field definitions, configuration options and usage information for the bom mode. The bom mode is used to import data around the following criteria:

- Create Items, Item Revisions and set/update their properties
- Apply Release Status
- Populate Item and ItemRevision Master forms
- Create/Update BOMs.

This mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=bom
```

5.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to

BOMs

FIELD	DESCRIPTION
ParentID or ItemID	Item ID of parent
RevID or ParentRev	Rev ID of parent
ChildID	Child Item ID
ChildRev	Child Rev ID
Qty	Quantity.
SeqNo	Sequence number for occurrence
New<FldName>	Supplies a new value for a field which has been used to identify occurrences to be update, i.e. NewSeqNo if SeqNo is used for identification.
SubstChildID	The Item ID of the Item to substitute the child
SubstChildRev	The Revision ID of the Revision to substitute the child
Predecessor	Item ID of predecessor part. (Used to provide sequencing/flow in BOM).
OccQty	Occurrence Quantity.

The following input data fields are valid in this mode and are specifically related to

Absolute Occurrences

FIELD	DESCRIPTION
AbsOccID	The Absolute Occurrence ID on the child occurrence.
OccType	The Occurrence Type on the child occurrence.
ContextItemID	Item ID (higher level) Assembly child is used in
ContextItemRevID	Rev ID (higher level) Assembly child is used in
ContextAbsOccID	Absolute Occurrence ID on child used in Context (higher level) assembly
ContextParentAbsOccID	Absolute Occurrence ID of immediate parent to child in Context (higher level) assembly.

The following input data fields are valid in this mode and are specifically related to

Engineering Change

FIELD	DESCRIPTION
ECID	The Absolute Occurrence ID on the child occurrence.
ECRev	The Occurrence Type on the child occurrence.
ECName	Item ID (higher level) Assembly child is used in
ECDesc	Rev ID (higher level) Assembly child is used in
ECType	Absolute Occurrence ID on child used in Context (higher level) assembly

The following input data fields are valid in this mode and are specifically related to

Occurrence Notes

FIELD	DESCRIPTION
ON:<Occ. note>	To set an occurrence note. This must be the internal definition, not the display names.

NOTE: Any occurrence notes which do not exist as types will be created as occurrence note types.

5.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to

BOMS

OPTION	DESCRIPTION
BOM FORMAT	Only “parent_child” is supported
VIEW TYPE	The BOMViewRevision Type
SET BOMS PRECISE	ON|OFF flag to control setting BOMS precise. (Default = OFF, imprecise)
CREATE BOMS	ON|OFF flag to control creation of BVRs
UPDATE BOMS	ON|OFF flag to control update of BVRs
BOM UPDATE OPTION	BOMViewRevision update from:
ADD OCCURRENCES	Adds occurrences to existing BOMViewRevision
REPLACE ALL OCCURRENCES	Removes all occurrences from the BVR before adding occurrences.
UPDATE OCCURRENCES	Update individual occurrences, but only the first one if multiple matches are found.
UPDATE ALL OCCURRENCES	Update individual occurrences and update ALL matching occurrences.
BOM UPDATE ID FIELDS	A comma separated list of fields to use to identify occurrences to update. For example, to match on ChildID and SeqNo; **ChildID,SeqNo**
ALTERNATES LIST SEPARATOR	Specify a different separator than a comma when supplying a list of alternate items
BOM UPDATE ADD OCCURRENCES	ON|OFF flag to control if new occurrences should be created
BOM UPDATE PRE DELETE ATTR	Defines a BOM attribute to use to find occurrences to be deleted before any updates for the BOM are done
BOM UPDATE PRE DELETE VALUE	Defines the value for the PRE DELETE ATTR to be matched
BOM UDPATE DELETE FIELD	Defines a field header name to be used to delete Occurrences. For example, the field name is user definable, i.e. “DELETE THIS”
BOM UPDATE DELETE VALUE	Defines the value for the DELETE FIELD to be matched to identify occurrences to be deleted. For example, “DELETE”
BOM SUBST CHILD OPTION	REVISE|SUBST flag to control BOM Child substation behaviour
BOM REVISE CHILD USE EXISTING	ON|OFF flag to control if existing Items/Revisions can be used in REVISE mode.
BOM REV RULE	Revision Rule to use. (Default = Latest Working).

NOTE: If UPDATE BOMS = ON is set the BOMViewRevision will toggle to satisfy the setting for SET BOMS PRECISE. I.e. if SET BOMS PRECISE = ON is set and the BOMVewRevision is not precise, it will be toggled to Precise and vice-versa.

NOTE: The following configuration options are only used if `BOM UPDATE OPTION` is set to `UPDATE OCCURRENCES` or `UPDATE ALL OCCURRENCES`:

1. BOM UPDATE ADD OCCURRENCES
2. BOM UPDATE PRE DELETE ATTR
3. BOM UPDATE PRE DELETE VALUE
4. BOM UPDATE DELETE FIELD
5. BOM UPDATE DELETE VALUE

The following configuration options are valid in this mode and are specifically related to

`Engineering Change`

OPTION	DESCRIPTION
`CREATE EC ITEMS`	ON|OFF flag to control creation of EC Items
`UPDATE EC ITEMS`	ON|OFF flag to control updating EC Items
`CREATE EC REVS`	ON|OFF flag to control creation of EC Revisions
`UPDATE EC REVS`	ON|OFF flag to control updating EC Revisions
`EC TYPE`	EC Item Type to create new EC Items as
`ADD CHILD TO EC FOLDER`	Name of defined EC ItemRevision psuedofolder to copy the Child ItemRevision into. For example, problem_items
`SET EC IN BOM IC CONTEXT`	Set the EC ItemRevision as the Incremental Change Context target in PSE when the BOM is loaded.
`EC FORMS LIST TO ADD`	A comma separated list of Form Types defined for the EC Type to add to new EC Item Revisions. Use the site preference defining the Relation to attach the forms in.
`EC PROCESS AUTO INIT`	ON|OFF flag to control whether a workflow process should be started using the EC process template defined in EC PROCESS TEMPLATE. (Default = OFF).
`EC PROCESS TEMPLATE`	Process Template defined for the EC Type.
`EC PROCESS DESC`	Global Process description added when creating workflow Processes.

NOTE: **`items`** mode can be used to set all Item and ItemRevision properties on EC Items/ItemRevisions.

5.3 MODULE NOTES

The following points should be considered when using this mode:

1. It is recommended the data be sorted by the ParentID and ParentRev fields and a parts list for a single parent should be wholly contained within the data file, otherwise there is a possibility any children for that parent in a subsequent data file may not be added under certain circumstances.

2. To control the creation of parent and child Items and Revisions use the options CREATE ITEMS and CREATE REVS to provide any combination of PARENT and CHILD as the value. For example:

```
CREATE ITEMS       = PARENT    - create parent items
CREATE ITEMS       = CHILD           - create child items
CREATE ITEMS       = PARENT,CHILD    - create parent and child items
CREATE REVS        = PARENT          - create parent revs
CREATE REVS        = CHILD           - create child revs
CREATE REVS        = PARENT,CHILD    - create parent and child revs
```

5.3.1 FULL OCCURRENCE UPDATE

This allows updating, deleting and adding of occurrences.

Any BOM attribute can be updated. Occurrences can be identified by one or a multiple of attributes, for example just ChildID or ChildID and SeqNo if there are multiple occurrences of the same child. By default occurrences will be matched just on the ChildID.

Any occurrences found will be updated with all supplied values. If a field is to be used to identify an occurrence and this value is also to be updated then a second field should be supplied with the same header name, but prefixed with "New". For example if matching on "SeqNo" then a field with the header "NewSeqNo" should be used to supply the new Sequence Number.

Several occurrences can be pre-deleted by specifying a particular attribute and value to look for. All occurrences with this value will be deleted. Otherwise occurrences can be deleted by 'marking' children in the data file. This is done by specifying a particular field and value in the data file.

If an occurrence is not marked for delete, but cannot be found for update then it will be added. However, there is an option to stop this if required. All options are configurable.

When identifying occurrences a [*] can be used in an identifying field to match any value, or a value of [NULL] can be used to match empty values. Thus it would be possible, for example, to unset the Sequence Number for ALL occurrences using a data file simply containing:

```
!~ParentID~ChildID~SeqNo
Assembly123~*~NULL
```

5.3.2 CHILD SUBSTITUTION

In update mode children can now be substituted with either existing Items or Revisions, or with Revised or Saved-As Revisions of the child.

Occurrences should be located as per the configuration option BOM OCCURRENCE UPDATE, which by default is to match on just the ChildID field.

By setting BOM UPDATE OPTION = UPDATE ALL OCCURRENCES all found occurrences will have their Item/Revision substituted.

All occurrences where substitution is done will retain all their existing occurrence attributes. However, new values for any attribute can be supplied in the data file in the normal way. If occurrence attributes are NOT to be kept then do not use these substitute options, but instead, delete the existing occurrence using a delete field and add the replacement child as a new occurrence.

This functionality uses the SubstChildID and SubstChildRev data fields along with BOM SUBST CHILD OPTION and BOM REVISE CHILD USE EXISTING configuration options.

1. **Substitution Mode – BOM SUBST CHILD OPTION = SUBST**

   Substitution mode is used to substitute an occurrence's Item or Revision with an existing Item or Revision. If the replacement Item or Revision is not found a NOT FOUND error message will be displayed. For imprecise BOMs only the SubstChildID is required, although supplying the SubstChildRev will not cause a problem. For precise BOMs, though, the Revision must be supplied too.

2. Substitution Mode – BOM SUBST CHILD OPTION = SUBST

Revise mode is used to either Revise the current occurrences' Revision to create a new Revision in the same Item or perform a Save As operation on the occurrences' revision to create a new item. This is controlled by whether the Item ID supplied in `SubstChildID` is the same as in `ChildID`. If they are the same a Revise is done, if they are not the same then a Save As is performed. If the `ChildRev` is not supplied then the Latest Revision of the occurrence will be Revised or Saved-As. If the `SubstChildRev` does supply a new Revision then the system will assign the new Revision, just as if doing an Assign when creating a revision in the interface.

If the Item supplied by `SubstChildID` for a Save-As operation already exists, or a specified Revision for a Revise operation already exists then by default an EXISTS error will be given. If, however, the existing Item or Revision is to be used to substitute the occurrence Revision, then set the configuration option `BOM REVISE CHILD USE EXISTING = ON`.

5.4 COMMAND LINE ARGUMENTS

The following command line arguments are valid in this mode and are specifically related to **BOMs**

FIELD	DESCRIPTION
`-bf=<`*`BOM Format`*`>`	BOM Format option to use. Currently only one option value is available: **parent_child**. TCENG PREFERENCE equivalent: **`IPS_IMP_Bom_Format`** Configuration Option equivalent: **`BOM FORMAT`**
`-vt=<`*`View Type`*`>`	Specify the View Type to use. TCENG PREFERENCE equivalent: **`IPS_IMP_View_Type`** Configuration Option equivalient: **`VIEW TYPE`**

5.5 EXAMPLES

Examples for `bom` mode can be found in Appendix C.

6. FORMS MODE

This section outlines the input data field definitions, configuration options and usage information for the forms mode. The forms mode is used to import data around the following criteria:

- Create Items, Item Revisions and set/update their properties
- Populate Item and ItemRevision Master forms
- Create/Update Forms and attach them to an Item or Revision

This mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=forms
```

6.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to

`forms`

FIELD	DESCRIPTION
`ItemID`	Item ID to attach the form to
`RevID`	The Rev ID of the ItemID to attach the form to
`FormName`	The name of the form
`FormDesc`	The description of the form
`FormType`	The Form Type
`F:<Attr>`	A form attribute to populate. This must be the must be the real name from the database and not the display name.

6.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to **forms**

OPTION	DESCRIPTION
UPDATE FORMS	ON|OFF flag to control updating forms
CREATE FORMS	ON|OFF flag to control creation of forms
FORM TYPE	
RELATION NAME	The relation type name to use. For example, Spec, Manif, Ref, Req, IMAN_custom_relation
RELATION PARENT TYPE	The type of parent, either ITEM or REV.

6.3 MODULE NOTES

The following points should be considered when using this mode:

1. The forms mode should be avoided for just setting Item or ItemRevision Master form attributes. The items mode should be used for these forms.

6.4 COMMAND LINE ARGUMENTS

The following command line arguments are valid in this mode and are specifically related to **forms**

FIELD	DESCRIPTION
-ft=<*Form Type*>	Specify default Form Type to use. TCENG PREFERENCE equivalent: **IPS_IMP_Form_Type** Configuration Option equivalent: **FORM TYPE**
-ao=<*Attach Obj Type*>	Attach to Object Type. Valid options are ITEM|REV. Configuration Option equivalent: **RELATION PARENT TYPE**
-rel=<*Relation Name*>	Specify the Relation to use where Relation is one of: Spec | Manif | Req | Ref. Configuration Option equivalent: **RELATION NAME**

6.5 EXAMPLES

Examples for forms mode can be found in Appendix D.

7. LOVS MODE

This section outlines the input data field definitions, configuration options and usage information for the list of values (LOVS) mode. The LOVS mode is used to import data around the following criteria:

- Populate/Re-populate existing LOVS

This mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=create_lovs
```

7.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to

`lovs`

FIELD	DESCRIPTION
`LovName`	The name of the LOV to populate
`LovType`	The type of LOV, i.e. STRING, INT or DOUBLE
`LovVal`	The value to add

7.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to

`forms`

OPTION	DESCRIPTION
`LOV NAME`	The name of the LOV to populate
`LOV TYPE`	STRING|INT|DOUBLE flag to control the LOV Type

7.3 MODULE NOTES

The following points should be considered when using this mode:

1. The List of Values (LOV) must be created first in Teamcenter Engineering prior to using `ips_data_upload` to populate it.

2. The following is a mapping of the LOV types that can be populated in Teamcenter Engineering by using `ips_data_upload`:

STRING	=	ListOfValuesString
INT	=	ListOfValuesInteger
DOUBLE	=	ListOfValuesDouble

7.4 COMMAND LINE ARGUMENTS

The following command line arguments are valid in this mode and are specifically related to **`lovs`**

FIELD	DESCRIPTION
`-lov=<`*`LOV Name`*`>`	Name of the LOV to populate. Configuration Option equivalent: **`LOV NAME`**
`-lt=<`*`LOV Type`*`>`	Specify the LOV Type of Data where LOV type is one of: STRING | INT | DOUBLE. Configuration Option equivalent: **`LOV TYPE`**

7.5 EXAMPLES

Examples for `create_lovs` mode can be found in Appendix E.

8. RELATIONS MODE

This section outlines the input data field definitions, configuration options and usage information for the relations mode. The relations mode is used to import data around the following criteria:

- Attach Items, Revisions, datasets or forms to Items or Revisions with a specified relation
- Move attachments from one relation to another relation within an Item or Revision

This mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=relations
```

8.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to

`relations`

FIELD	DESCRIPTION
`ItemID or ParentID`	The Item ID of the parent Item
`[RevID or ParentRev]`	OPTIONAL. The Revision ID of the Parent Revision
`ChildID*`	The ItemID of an item to paste in
`[ChildRev]`	OPTIONAL. The Revision ID of the Revision to paste into the parent.
`FormName*`	The name of the form to paste into the parent
`DsetName*`	The name of the dataset to paste into the parent
`DsetType`	The Type of the dataset if a dataset is to be pasted into the parent
`RelationName`	The Name/Type of the relation to paste as
`[NewRelation]`	OPTIONAL. New relation name/type to move the child object to. For example, from `RelationName` to `NewRelation`.

* Only one of the fields; ChildID, FormName and DsetName can be used. These fields are mutually exclusive.

8.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to **`relations`**

OPTION	DESCRIPTION
`RELATION NAME`	The Name/Type of the relation to paste as
`RELATION PARENT TYPE`	The type of parent object ITEM or REV. If REV is used and no ParentRev or RevID is given in the data file then the latest revision will be used.
`RELATION CHILD TYPE`	The Type of child object ITEM or REV. If REV is used and no ChildRev ID is given in the data file then the latest revision will be used.
`NEW RELATION NAME`	Optional new relation name/type to move the child object to. For example, from RelationName to NewRelation

8.3 MODULE NOTES

The following points should be considered when using this mode:

1. If ChildID, RevID, DsetName and DsetType are all supplied then the dataset will be found in the specified child revision and copied to the parent Revision or item Accordingly.

2. Both Parent and Child objects must already exist in Teamcenter Engineering. This mode will not create the objects.

3. The following order of precedence is used for specifying a relation:

    a. The Internal Teamcenter Engineering Relation Name
    b. One of the following Keywords:

        i. SPEC (IMAN_specification)
        ii. MANIF (IMAN_manifestation)
        iii. REQ (IMAN_requirement)
        iv. REF (IMAN_reference)

8.4 COMMAND LINE ARGUMENTS

The following command line arguments are valid in this mode and are specifically related to **`relations`**

FIELD	DESCRIPTION
`-rel=<`*`Relation Name`*`>`	Name of the Relation to populate. Configuration Option equivalent: **`RELATION NAME`**
`-newrel=<`*`New Relation Name`*`>`	Specify the new Relation Name to populate. Configuration Option equivalent: **`NEW RELATION TYPE`**

8.5 EXAMPLES

Examples for `relations` mode can be found in Appendix F.

9. DATASETS MODE

This section outlines the input data field definitions, configuration options and usage information for the datasets mode. The datasets mode is used to import data around the following criteria:

- Create Datasets/Update existing datasets' properties
- Import files to datasets
- Update existing files or replace existing Named References
- Attach the datasets to Items/Revisions in a specified relation
- Create Items, Item Revisions and set/update their properties

This mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=datasets
```

9.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to

`datasets`

FIELD	DESCRIPTION
`ItemID`	The Item ID of the parent Item
`[RevID]`	OPTIONAL. The Revision ID of the Parent Revision
`DsetName*`	The name of the dataset to paste in
`DsetType*`	The type of the dataset
`DsetDesc*`	Dataset Description
`DsetGroup`	Dataset Owning Group
`DsetOwner*`	Dataset Owning User
`DsetCrtDate*`	Dataset Create Date
`DsetModDate*`	Dataset Mod Date
`DsetModBy*`	Dataset Last Modified By User
`NewDsetName`	New Dataset Name. Use DsetName to find the dataset and NewDsetName to change the name of the dataset.
`Volume`	Defines the volume for new files imported
`DsetID`	Dataset ID
`DsetRev`	Dataset Revision

* These fields can be replaced with the standard item header fields, but if creating items at the same time then use the Dset header fields if the dataset is to have different data to that of the item and rev.

The following input data fields are valid in this mode and are specifically related to

`files`

FIELD	DESCRIPTION
`FileRef`	The name of the dataset reference to attach to
`File`	The name and path of the file to import

The following input data fields are valid in this mode and are specifically related to

`forms`

FIELD	DESCRIPTION
`FormName`	The Name of a form to add as a reference to the dataset
`FormType`	The Type of the form
`FormRef`	The name of the dataset reference to attach to
`F:<attr>`	The attribute name on the form to populate

The following input data fields are valid in this mode and are specifically related to

`relations`

FIELD	DESCRIPTION
`RelationName`	The Name/Type of the relation to paste as

9.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to **datasets**

OPTION	DESCRIPTION
CREATE DATASET	ON|OFF flag to control creation of Datasets
UPDATE DATASET	ON|OFF flag to control updating of Datasets
DATASET TYPE	The default dataset type for new datasets
SET OWNER BY PARENT	ON|OFF flag to control owner settings
FILE REF NAME	Dataset reference to attach file to
UPDATE FILES	ON|OFF flag to control updating file references
UPDATE FILE REF NAME	Update file reference option from:
NEW REF	Create new references only, do nothing if exists
ADD REF	Add new reference, even one already exists
REPLACE REF	Replace existing ref
NEW VERSION	Create new version of dataset and replace
CREATE NEW DATASET VERSION	ON|OFF flag to be used instead of NEW VERSION option in UPDATE FILE REF NAME option. This can be used in conjunction with the 3 other options for more flexibility.
FILE REF REPLACE OPTION	Options to define REPLACE REF option from:
ALL REFS	Replace ALL references with new reference
MATCH FILE NAME	Replay only the reference with the same for type
MATCH REF TYPE	Replace ALL references with the reference type provided with the new file
FORM REF NAME	Dataset reference to attach form to
UPDATE FORMS	ON|OFF flag to control updating form references
UPDATE FORM REF NAME	Update file reference option from:
NEW REF	Create new references only, do nothing if exists
ADD REF	Add new reference, even one already exists
REPLACE REF	Replace existing reference
NEW VERSION	Create new version of form and replace
FORM REF REPLACE OPTION	Options to define REPLACE REF option from:
ALL REFS	Replace ALL references with new reference
MATCH FORM TYPE	Replay only the reference with the same for type
RELATION NAME	The Name/Type of the relation to paste as

RELATION PARENT TYPE	The type of parent object ITEM or REV. If REV is used and no ParentRev or RevID is given in the data file then the latest revision will be used.

NOTE: All items mode configuration options are available in this mode.

9.3 MODULE NOTES

The following points should be considered when using this mode:

1. For best performance sort the data file by ItemID, RevID, Dataset Name and Ref Name fields.

2. If the Revision ID is not supplied in the data file then datasets will be attached to the Item (Default) or the Latest Revision, depending on the setting of the configuration option RELATION PARENT TYPE.

9.4 COMMAND LINE ARGUMENTS

The following command line arguments are valid in this mode and are specifically related to **folders**

FIELD	DESCRIPTION
-ds_type=<*Dataset Type*>	Dataset Type. Configuration Option equivalent: **DATASET TYPE**
-rel=<*Relation Type*>	Specify the Relation to use where Relation is one of: Spec | Manif | Req | Ref. Configuration Option equivalent: **FORM TYPE**
-pt=<*Parent Type*>	Parent Object type to attach to. Configuration Option equivalent: **RELATION PARENT TYPE**

9.5 EXAMPLES

Examples for datasets mode can be found in Appendix G.

10. FOLDER MODE

This section outlines the input data field definitions, configuration options and usage information for the folders mode. The folders mode is used to import data around the following criteria:

- Create Folders with the ability to supply full folder paths
- Populate folders

This mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=folder_create
-m=folder_insert
```

10.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to

`folders`

FIELD	DESCRIPTION
`ItemID`	The ItemID of the Item to insert into the folder
`Folder`	The Folder Name or Path
`FolderDesc`	The folder Description
`FolderPath`	A Folder Path to create (nested folder structure).
`FolderType`	The Type of folder to use.

10.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to **folders**

OPTION	DESCRIPTION
FOLDER SEPERATOR	Define the character separator in folder path
TOP FOLDER	The Name of a TOP folder to create folders in
TOP FOLDER DESCRIPTION	The description of the top folder
DEFAULT FOLDER DESCRIPTION	Default description for new folders
PUT FAILS IN NEWSTUFF	ON|OFF flag to control using Newstuff if any folder inserts fail
GLOBAL FOLDER SEARCH	ON|OFF flag to control if the whole database is searched for a folder (ON) or just folders owned by the login user (OFF)
TOP FOLDER TYPE	The Type of the top folder. (Default is "Folder")
FOLDER TYPE	The Name/Type of the Folder

10.3 MODULE NOTES

The following points should be considered when using this mode:

1. If any folders cannot be found, they will be created.

2. For the best performance, sort the data file by folder path.

3. If no Folder Type is specified by the FOLDER TYPE configuration option or FolderType field a default Folder Type of “Folder” will be used.

4. If the configuration options TOP FOLDER = Home and TOP FOLDER TYPE = Folder, new Folder objects will be created/inserted into the users’ Home folder.

10.4 COMMAND LINE ARGUMENTS

The following command line arguments are valid in this mode and are specifically related to **`folders`**

FIELD	DESCRIPTION
`-tf=<TopFolder>`	Name of Top Folder to populate. Configuration Option equivalent: **`TOP FOLDER`**
`-fs=<Folder Seperator>`	Folder path separator. Single character separator. i.e. FolderA/FolderB/FolderC; Seperator = / Configuration Option equivalent: **`FOLDER SEPERATOR`**

10.5 EXAMPLES

Examples for `folders` mode can be found in Appendix H.

11. PROJECTS MODE

This section outlines the input data field definitions, configuration options and usage information for the projects mode. The projects mode is used to import data around the following criteria:

- Create, Update Projects and assign team members
- Assign V9+ Projects to Items and Revisions.

This mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=projects
```

NOTE: This functionality is only available in Teamcenter Engineering Version 9 and later.

11.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to

`projects`

FIELD	DESCRIPTION
`ProjID`	MANDATORY. Project ID
`ProjName`	Project Name. If not supplied then the name with be set with the Project ID
`ProjDesc`	ProjectDescription
`ProjMemberUser`	Project Member user. If used without role and group then all group members for the user will be assigned
`ProjMemberGroup`	Project Member Group. Can be supplied without role and user to assign just to a group.
`ProjMemberRole`	Project Member Role. If used without user and role then all group members with this role will be assigned.
`ProjAdminUser`	Project Admin User
`ProjPrivUser`	Project Privileged user. Can be a comma separated list of privileged users.
`ItemID`	The ID of the Item to be assigned to the project
`RevID`	The ID of the Revision to be assigned to the project

11.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to **`projects`**

OPTION	DESCRIPTION
`CREATE PROJECTS`	ON|OFF flag to control if Projects should be created if they do not exist
`UPDATE PROJECTS`	ON|OFF flag to control if Projects should be updated if they already exist
`PROJECT ASSIGN TO TYPE`	ITEM|REV. Controls if the specified ITEM or REVision should be assigned to the project. Assigning an item also assigns its Revisions automatically (this is standard TcEng bahaviour).
`PROJECT APPEND MEMBERS`	ON|OFF flag to control if Project members can be added (appended) to an existing project (team). Default = OFF. If not used, any existing project assigned team will be replaced!

11.3 MODULE NOTES

The following points should be considered when using this mode:

1. To assign Items or Revisions to a project then the login user must be a privileged user for the project.

2. Use separate data lines to assign multiple Items/Revisions to a project, one line per Item/Revision.

3. Issues may be encountered trying to delete any existing members which are not present in the new team being specified. It is recommended only to assign the team when the project is created.

4. Projects can be assigned in `items` mode.

11.4 COMMAND LINE ARGUMENTS

None.

11.5 EXAMPLES

Examples for `projects` mode can be found in Appendix I.

12. CLASSIFICATION MODE

This section outlines the input data field definitions, configuration options and usage information for the classificiation mode. The classification mode is used to import data around the following criteria:

- Create, Update or Delete ICOs (InClass Objects)
- Classify Items/Revisions

This mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=inclass
```

NOTE: This functionality is only available in Teamcenter Engineering Version 8 and later.

12.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to

`classification`

FIELD	DESCRIPTION
`IcsClassID`	MANDATORY. In-Class Class ID
`IcoPUID`	InClass PUID (Tag)
`IcoID`	Inclass ICO ID
`IcsAttr:` `<inclass_attr_id>`	Inclass Attribute ID. Use repeatedly to specify as many attribute IDs as required.
`ItemID`	The ID of the Item to be classified
`RevID`	The ID of the Revision to be classified

12.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to

`classification`

OPTION	DESCRIPTION
`INCLASS CREATE ICO`	ON|OFF flag to control if InClass ICOs should be created if they do not exist
`INCLASS UPDATE ICO`	ON|OFF flag to control if InClass ICOs should be updated if they already exist

`INCLASS UPDATE ICO OPTION`
`SINGLE ICO ONLY`
`FIRST ICO`
`ALL ICOS`
	ICO update option from:
	Only singly classified objects can be updated
	If multiple ICOs on an object, only update the first
	Update all ICOs on an object

`INCLASS DEFAULT CLASS ID`	<ClassID> Specifies a default class ID
`INCLASS ICO CLASSIFY OBJECT`	ITEM|REV. Controls if the specified ITEM or REVision should be classified with the InClass ICO
`INCLASS ALLOW MULTIPLE CLASSIFICATION`	ON|OFF flag to control if multiple ICOs are allowed to be classified on a single item or revision. If OFF only one ICO classification is allowed. This option only affects the creation of ICOs and subsequent classification. So if the item, or rev, is already classified and this option is OFF then the ICO will not be created

12.3 MODULE NOTES

The following points should be considered when using this mode:

1. Use standard `items` mode fields to specify Item and Revision details, in particular ItemID and RevID to specify the Item ID and optionally the Revision ID of the Item or Revision to classify. All other 'items' fields can be used to create or update items and revs too.

2. The Class ID can be defaulted in the configuration file or provided in a field.

3. If no IcoID or IcoPUID field is specified then the Item ID specified will be used as the IcoID, or 'ItemID/RevID' if classifying the revision.

4. To update individual ICOs on an object which has multiple classifications then use the IcoPUID.

5. To add multiple classification objects to one Item or Revision, ensure INCLASS UPDATE ICO is OFF, otherwise only existing ICOs will be updated.

6. By specifying a Class ID of 'UNCLASSIFY' an Item or Rev can be unclassified. This will remove ALL ICOs classifying an object.

7. By specifying a Class ID of 'DELETE' a specific ICO can be deleted. Any references (links) from objects (i.e. Workspace Objects) classified by the specified ICO will be deleted, but the Workspace Object will remain, before the ICO is deleted. The delete is based on specifying the ICOs PUID in the IcoPUID field.

12.4 COMMAND LINE ARGUMENTS

None.

12.5 EXAMPLES

Examples for `classification` mode can be found in Appendix J.

13. IDENTIFIERS MODE

This section outlines the input data field definitions, configuration options and usage information for the identifiers mode. The identifiers mode is used to import data around the following criteria:

- Create Alternate Identifiers and Identifier Revisions
- Update Alternate Identifiers and Identifier Revisions

This mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=identifiers
```

NOTE: This functionality is only available in Teamcenter Engineering Version 9 and later.

13.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to **`identifiers`**

FIELD	DESCRIPTION
`AlternateID`	Alternate ID
`AlternateRevID`	Alternate ID Supplementary ID for Revision
`AlternateIDContext`	ID Context to use – must be defined in Teamcenter Engineering
`AltIDName`	Alternate ID object Name
`AltIDDesc`	Alternate ID object Description
`AltRevIDName`	Alternate Revision Name
`AltRevIDDesc`	Alternate Revision description
`ItemID`	ItemID the Identifier object is to be attached to
`RevID`	RevID the supplementary Identifier object is to be attached to
`AltIDOwner`	Owning user
`AltIDGroup`	Owning group

13.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to

`identifiers`

OPTION	DESCRIPTION
`CREATE IDENTIFIERS`	ON|OFF flag to control if Identifiers should be created if they do not exist
`UPDATE IDENTIFIERS`	ON|OFF flag to control if Identifiers should be updated if they already exist
`ALTERNATE ID CONTEXT`	Default ID Context to use when creating Alternate IDs.
`CREATE IDFR REVS`	ON|OFF flag to control if IdentifierRev objects should be created if they do not already exist at the Supplied Revision. Default = ON
`UDPATE IDFR REVS`	ON|OFF flag to control if IdentifierRev objects should be updated if they do already exist at the Supplied Revision. Default = OFF

13.3 MODULE NOTES

The following points should be considered when using this mode:

1. Use standard `items` mode fields to specify Item and Revision details in particular `ItemID` and `RevID` to specify the Item and Revision to attach the Alternate ID and Supplemental Identifier to.

2. The `AlternateID` and `AlternateRevID` fields are mandatory requirements.

3. The Alternate ID Context must be supplied in either a data field or configuration option. This is also a mandatory requirement.

4. The input fields Name and Desc will be used if no `AltIDName` and `AltIDDesc` fields are specified. The input fields `RevName` and `RevDesc` will be used if no `AltRevIDName` and `AltRevIDDesc` fields are specified.

5. If no Name field is specified at all, the default Identifier object name will be the same as the ID.

6. Other cfg options, `CREATE ITEMS`, `CREATE REVS`, `UPDATE ITEMS`, `UPDATE REVS`, `UPDATE ITEM ATTACHMENTS` etc. can be used in conjunction with creating and updating Identifier objects.

7. When working in other modes (i.e. datasets), if `CREATE` or `UPDATE IDENTIFIERS` is set appropriately, system properties will be propagated to Identifier objects.

13.4 COMMAND LINE ARGUMENTS

None.

13.5 EXAMPLES

Examples for `identifiers` mode can be found in Appendix K.

14. MANUFACTURING MODE

This section outlines the input data field definitions, configuration options and usage information for the manufacturing (mfg) mode. The manufacturing mode is used to import data around the following criteria:

- Create and Update MEProcess, MEOP and MEWorkArea objects
- Build MEProcess/MEOP Bill of Process (BOP)
- Create Activities
- MEActivity and MEOP sequencing (PERT charts)
- Consume Product and Plant into Process BOM

This mode is specified in the command line options for the utility or by setting the command line argument:

```
-m=mfg
```

NOTE: This functionality is only available in Teamcenter Engineering Version 9 and later.

14.1 INPUT DATA FIELDS

The following input data fields are valid in this mode and are specifically related to

`Manufacturing`

FIELD	DESCRIPTION
`ProcID`	MEProcess ID
`ProcRevID`	MEProcess Revision ID
`ProcType`	MEProcess class Type
`ProcName`	MEProcess Name
`ProcDesc`	MEProcess Description
`OpID`	MEOP ID
`OpRevID`	MEOP Revision ID
`OpType`	MEOP Class Type
`OpName`	MEOP Name
`OpDesc`	MEOP Description
`WorkID`	MEWorkArea ID
`WorkRevID`	MEWorkArea Revision ID
`WorkType`	MEWorkArea Class Type
`WorkName`	MEWorkArea Name
`WorkDesc`	MEWorkArea Description

ActType	MEActivity Class Type
ActName	MEActivity Name
ActDesc	MEActivity Description
ActTime	MEActivity time (Duration)
ActPreds	Name of MEActivity Objects attached to MEOP which are predecessor Activities. (Used to provide sequencing/flow in MSE).
FormType	MEActivity Form Type
FormName	MEActivity Form Name
FormDesc	MEActivity Form Description
F:<attr>	MEActivity Form attribute to populate. This must the real attribute name, not the display name.
PlantTargetID	Item ID for Plant BOM to link/consume from
PlantTargetRevID	Rev ID for Plant BOM to link/consume from
ProductTargetID	Item ID for Product BOM to link/consume from
ProductTargetRevID	Rev ID for Plant BOM to link/consume from
TargetChildSeqNo	Occurrence Seq No in Product/Plant BOM
TargetChildID	Occurrence Item ID in Product/Plant BOM
TargetChildRev	Occurrence Rev ID in Product/Plant BOM
TargetChildAbsOccID	Occurrence AbsOccID in Product/Plant BOM context
ProcChildID	Occurrence Item ID of operation in Process BOM to consume target occurrence
ProcChildRev	Occurrence Rev ID of operation in Process BOM to consume target occurrence

14.2 CONFIGURATION OPTIONS

The following configuration options are valid in this mode and are specifically related to **`manufacturing`**

OPTION	DESCRIPTION
CREATE PROCESSES	ON|OFF flag to control creating MEProcesses if they do not exist
UPDATE PROCESSES	ON|OFF flag to control updating MEProcesses if the already exist.
CREATE PROCESS REVS	ON|OFF flag to control creating MEProcess Revisions if they do not exist
UPDATE PROCESS REVS	ON|OFF flag to control updating MEProcess Revisions if the already exist.
CREATE OPERATIONS	ON|OFF flag to control creating MEOPs if they do not exist
UPDATE OPERATIONS	ON|OFF flag to control updating MEOPs if the already exist.
CREATE OPERATION REVS	ON|OFF flag to control creating MEOP Revisions if they do not exist
UPDATE OPERATION REVS	ON|OFF flag to control updating MEOP Revisions if they already exist
CREATE ACTIVITIES	ON|OFF flag to control creating MEActivities if they don't already exist
UPDATE ACTIVITIES	ON|OFF flag to control updating MEActivities if they already exist
CREATE ACTIVITY FORMS	ON|OFF flag to control creating forms if they do not exist on an MEActivity.
UPDATE ACTIVITY FORMS	ON|OFF flag to control updating forms if they already exist on an MEActivity.
LINK OP TO PROCESS	ON|OFF flag to control if Process BOM (MEProcess← MEOP(s)) should be created and/or updated for the MEProcess structure (BOP). If OFF objects are created independently – no BOP structure
PROCESS TYPE	A valid MEProcess Type to use
OPERATION TYPE	A valid MEOP Class Type to use
ACTIVITY TYPE	A valid MEActivity Type to use
ACTIVITY FORM TYPE	A valid Form Type to use
DEFAULT OWNER	Default Owning User. Must be a valid user
DEFAULT GROUP	Default Owning Group. Must be a valid group

CREATE WORKAREAS	ON|OFF flag to control if MEWorkArea objects should be created if they do not exist
UPDATE WORKAREAS	ON|OFF flag to control if MEWorkArea objects should be updated if they already exist.
CREATE WORKAREA REVS	ON|OFF flag to control if MEWorkArea Revisions should be created if they do not exist
UPDATE WORKAREA REVS	ON|OFF flag to control if MEWorkArea Revisions should be updated if they already exist
WORKAREA TYPE	MEWorkArea Class Type. The valid name of MEWorkArea Type to use.
LINK PRODUCT TO PROCESS	ON|OFF flag to control linking Product structure root to Process structure root.
LINK PLANT TO PROCESS	ON|OFF flag to control linking Plant structure root to Process structure root.

14.3 MODULE NOTES

The following points should be considered when using this mode:

1. Processes, Operations and WorkAreas can have their system properties set by running in items mode after they have been created.

2. If `PROCESS TYPE` configuration option and `ProcType` data field are not set, the Process Type will default to `MEProcess`. The data field value takes precedence over the configuration option.

3. If `OPERATION TYPE` configuration option and `OpType` data field are not set, the Operation Type will default to `MEOP`. The data field value takes precedence over the configuration option.

4. If `ACTIVITY TYPE` configuration option and `ActType` data field are not set, the Activity Type will default to `MEActivity`. The data field value takes precedence over the configuration option.

5. The `DEFAULT OWNER/DEFAULT GROUP` options apply to all objects in mfg mode and will be applied at create or update (if configuration allows) of an object.

6. The `DEFAULT OWNER/DEFAULT GROUP` is propagated to the Process, Process Master form, Process Revision, Process Revision master form and BVs/BVRs. To change other attachments, please use other modes. This is the same for Operations.

7. See Section 2 for BOM module config settings/input fields. The BOM module functionality is fully exposed to the MFG module and used to create Process<-- Operation structures.

8. All supplied object Types are validated before attempting to create objects.

9. Activities are Created in the context of an Operation. To create activities, the Operation ID/Rev must be supplied.

10. If `UPDATE ACTIVITY = OFF`, any form updates are disabled no matter what `ACTIVITY FORM` settings are present. If `ACTIVITY UPDATE = ON,` the `ACTIVITY FORM` configuration options are then activated.

11. All mfg objects are created BEFORE being added to the Process structure.

12. Assumption: There will be only 1 form of a particular type with a specific name attached to an Activity.

13. Use mfg mode to create all MEWorkArea objects and then use BOM mode to create a Plant structure with bom data set.

14. If `WORKAREA TYPE` configuration option and `WorkType` data field are not set, the WorkArea Type will default to `MEWorkArea`. The data field value takes precedence over the configuration option.

15. If relation of ACTIVITIES is set in 'REV RELEASE RELATIONS' configuration options list when applying a status to a MEOP Revision, the Activity_Root and all subsequent attachments/Activities etc. will also have the same status applied. This works recursively.

14.4 COMMAND LINE ARGUMENTS

None.

14.5 EXAMPLES

Examples for `forms` mode can be found in Appendix L.

15. APPENDIX A: DATE FORMATTING SEQUENCES

The following can be used with the **DATE FORMAT** configuration option.

%a locale's abbreviated weekday name (Sun..Sat)
%A locale's full weekday name, variable length (Sunday..Saturday)
%b locale's abbreviated month name (Jan..Dec)
%B locale's full month name, variable length (January..December)
%c locale's date and time (Sat Nov 04 12:02:33 EST 1989)
%C century (year divided by 100 and truncated to an integer) [00-99]
%d day of month (01..31)
%D date (mm/dd/yy)
%e day of month, blank padded (1..31)
%F same as %Y-%m-%d
%g the 2-digit year corresponding to the %V week number
%G the 4-digit year corresponding to the %V week number
%h same as %b
%H hour (00..23)
%I hour (01..12)
%j day of year (001..366)
%k hour (0..23)
%l hour (1..12)
%m month (01..12)
%M minute (00..59)
%n a newline
%N nanoseconds (000000000..999999999)
%p locale's upper case AM or PM indicator (blank in many locales)
%P locale's lower case am or pm indicator (blank in many locales)
%r time, 12-hour (hh:mm:ss [AP]M)
%R time, 24-hour (hh:mm)
%s seconds since `00:00:00 1970-01-01 UTC' (a GNU extension)
%S second (00..60); the 60 is necessary to accommodate a leap second
%t a horizontal tab
%T time, 24-hour (hh:mm:ss)
%u day of week (1..7); 1 represents Monday
%U week number of year with Sunday as first day of week (00..53)
%V week number of year with Monday as first day of week (01..53)
%w day of week (0..6); 0 represents Sunday

%W week number of year with Monday as first day of week (00..53)

%x locale's date representation (mm/dd/yy)

%X locale's time representation (%H:%M:%S)

%y last two digits of year (00..99)

%Y year (1970...)

%z RFC-822 style numeric timezone (-0500) (a nonstandard extension)

%Z time zone (e.g., EDT), or nothing if no time zone is determinable

16. APPENDIX B: ITEMS MODULE EXAMPLES

To Create or update items, setting owner, group, create date and Project ID on the revision master form. Put any new items into the folder "Upload Tests".

Config File

CREATE ITEMS = ON

UPDATE ITEMS = ON

CREATE REVS = ON

UPDATE REV S = ON

DATE FORMAT = %d/%m/%Y

FOLDER NAME = Upload Tests

DEFAULT REV = A

ITEM TYPE = Item

DATE FORMAT = %d/%m/%Y

Data File

!~ItemID~RevID~Owner~Group~CreateDate~RM:project_id~RMD:date_valid_from

BomAssy1~A~williams~designers~16/03/1997~Project X~01/04/1997

BomChild1~A~williams~designers~16/03/1997~Project X~01/04/1997

BomChild2~A~williams~designers~16/03/1997~Project X~01/04/1997

BomChild3~A~williams~designers~16/03/1997~Project X~01/04/1997

To Perform a Check-Out on an ItemRevision and all attachments that can have a Check-Out set. No Item/Item Attachmnents are checked! Any objects that have an existing Check-Out set will be Checked-In – these objects will NOT have the Check-Out set.

Config File

CREATE ITEMS = ON

UPDATE ITEMS = OFF

CREATE REVS = ON

UPDATE REVS = ON

UPDATE REV ATTACHMENTS = ON

UPDATE ITEM ATTACHMENTS = OFF

UPDATE FORMS = ON

UPDATE DATASET = ON

UPDATE BOMS = ON

#New CICO config options

CREATE CICO = ON

UPDATE CICO = CHECK IN

CHECK OUT REASON = Guy's Check-Out test

Data File

!~ItemID~RevID

GUY98765~A

To Perform a Checkin-On all Checked-Out objects attached to an ItemRevision. No Item/Item Attachments are checked! The Change_id is supplied in the data field. No existing Check-Outs will be changed.

Config File

CREATE ITEMS = ON

UPDATE ITEMS = OFF

CREATE REVS = ON

UPDATE REVS = ON

UPDATE REV ATTACHMENTS = ON

UPDATE ITEM ATTACHMENTS = OFF

UPDATE FORMS = ON

UPDATE DATASET = ON

UPDATE BOMS = ON

#New CICO config options

CREATE CICO = ON

UPDATE CICO = OFF

CHECK OUT REASON = Guy's Check-Out test

Data File

!~ItemID~RevID~CheckOutID

GUY98765~A~CID_982KDME8001

17. APPENDIX C: BOM MODULE EXAMPLES

To create a basic BOM, set the Quantity in the BOM for each occurrence added and set owner, group and CreateDate properties on the Items.

Config File

MODE = bom

BOM FORMAT = parent_child

DATE FORMAT = %d/%m/%y %H:%M:%S

FOLDER NAME = Upload Tests

DEFAULT REV = A

SET BOMS PRECISE = OFF

Data File

ITEM TYPE = Item

VIEW TYPE = Design

DATE FORMAT = %d/%m/%Y

BOM FORMAT = parent_child

!~ItemID~ChildID~Qty~Owner~Group~CreateDate

BomAssy1~BomChild1~2~williams~designers~16/03/1997

BomAssy1~BomChild2~3~williams~designers~16/03/1997

BomAssy1~BomChild3~5~williams~designers~16/03/1997

Blank all sequence numbers in a BOM...

Config File

CREATE BOMS = ON

UPDATE BOMS = ON

BOM UPDATE OPTION = UPDATE ALL OCCURRENCES

Data File

!~ParentID~ChildID~SeqNo

Assembly123~*~NULL

Reset sequence numbers (from 10,20,30... to 1,2,3...), matching occurrences on ChildId and SeqNo...

Config File

BOM UPDATE OPTION = UPDATE OCCURRENCES

BOM UPDATE ID FIELDS = ChildID,SeqNo

Config File

!~ParentID~ChildID~SeqNo~NewSeqNo

Assembly123~Child1~10~1

Assembly123~Child2~30~2

Assembly123~Child3~30~3

To Pre-delete occs with Occ Note "RDN Letter" set and then update occurrences by matching on ChildID and SeqNo. Update Qty and RDN Letter, do NOT add any occurrences which are not found and delete occurrences where RDN Letter = X.

Config File

BOM UPDATE OPTION = UPDATE OCCURRENCES

BOM UPDATE ID FIELDS = ChildID,SeqNo

BOM UPDATE ADD OCCURRENCES = OFF

BOM UPDATE PRE DELETE ATTR = RDN Letter

BOM UPDATE PRE DELETE VALUE = *

BOM UPDATE DELETE FIELD = RDN Letter

BOM UPDATE DELETE VALUE = X

Data File

!~ParentID~ChildID~SeqNo~Qty~ON:RDN Letter

Assembly123~Child1~1~2~C

Assembly123~Child2~2~20~R

Assembly123~Child3~3~5~C

Assembly123~Child4~4~0~X

Assembly123~Child5~5~0~X

Delete ALL occurrences in Assembly123…

Config File

BOM UPDATE OPTION = UPDATE OCCURRENCES

BOM UPDATE ADD OCCURRENCES = OFF

BOM UPDATE PRE DELETE ATTR = ChildID

BOM UPDATE PRE DELETE VALUE = *

Data File

!~ParentID~ChildID

Assembly123~*

Set InContext AbsOccID in Parent Assembly XSUPERCAR...

Config File

BOM FORMAT = parent_child

UPDATE BOMS = ON

CREATE BOMS = ON

BOM UPDATE OPTION = UPDATE OCCURRENCES

BOM UPDATE ID FIELDS = ChildID,AbsOccID,SeqNo

BOM REV RULE = RR Latest Agreed Scenery

Data File

!~ParentID~ParentRev~ChildID~ChildRev~SeqNo~AbsOccID~ContextItemID~ContextItemRevID~ContextAbsOccID~ContextParentAbsOccID

XSUPERCAR~A~XCAR99~A~10~Vectra~~~~

XCAR99~A~XAXLE99~A~10~RearAxle~XSUPERCAR~A~Vectra-RearAxle~Vectra

XAXLE99~A~XWHEEL~A~10~LeftWheel~XSUPERCAR~A~Vectra-RearAxle-LeftWheel~Vectra-RearAxle

XWHEEL~A~XWHEELNUT~A~10~nut1~XSUPERCAR~A~Vectra-RearAxle-LeftWheel-nut1~Vectra-RearAxle-LeftWheel

XWHEEL~A~XWHEELNUT~A~20~nut2~XSUPERCAR~A~Vectra-RearAxle-LeftWheel-nut2~Vectra-RearAxle-LeftWheel

XWHEEL~A~XWHEELNUT~A~30~nut3~XSUPERCAR~A~Vectra-RearAxle-LeftWheel-nut3~Vectra-RearAxle-LeftWheel

XWHEEL~A~XWHEELNUT~A~40~nut4~XSUPERCAR~A~Vectra-RearAxle-LeftWheel-nut4~Vectra-RearAxle-LeftWheel

XAXLE99~A~XWHEEL~A~20~RightWheel~XSUPERCAR~A~Vectra-RearAxle-RightWheel~Vectra-RearAxle

XWHEEL~A~XWHEELNUT~A~10~nut1~XSUPERCAR~A~Vectra-RearAxle-RightWheel-nut1~Vectra-RearAxle-RightWheel

XWHEEL~A~XWHEELNUT~A~20~nut2~XSUPERCAR~A~Vectra-RearAxle-RightWheel-nut2~Vectra-RearAxle-RightWheel

XWHEEL~A~XWHEELNUT~A~30~nut3~XSUPERCAR~A~Vectra-RearAxle-RightWheel-nut3~Vectra-RearAxle-RightWheel

XWHEEL~A~XWHEELNUT~A~40~nut4~XSUPERCAR~A~Vectra-RearAxle-RightWheel-nut4~Vectra-RearAxle-RightWheel

XCAR99~A~XAXLE99~A~20~FrontAxle~XSUPERCAR~A~Vectra-FrontAxle~Vectra

XAXLE99~A~XWHEEL~A~10~LeftWheel~XSUPERCAR~A~Vectra-FrontAxle-LeftWheel~Vectra-FrontAxle

XWHEEL~A~XWHEELNUT~A~10~nut1~XSUPERCAR~A~Vectra-FrontAxle-LeftWheel-nut1~Vectra-FrontAxle-LeftWheel

XWHEEL~A~XWHEELNUT~A~20~nut2~XSUPERCAR~A~Vectra-FrontAxle-LeftWheel-nut2~Vectra-FrontAxle-LeftWheel

XWHEEL~A~XWHEELNUT~A~30~nut3~XSUPERCAR~A~Vectra-FrontAxle-LeftWheel-nut3~Vectra-FrontAxle-LeftWheel

XWHEEL~A~XWHEELNUT~A~40~nut4~XSUPERCAR~A~Vectra-FrontAxle-LeftWheel-nut4~Vectra-FrontAxle-LeftWheel

XAXLE99~A~XWHEEL~A~20~RightWheel~XSUPERCAR~A~Vectra-FrontAxle-RightWheel~Vectra-FrontAxle

XWHEEL~A~XWHEELNUT~A~10~nut1~XSUPERCAR~A~Vectra-FrontAxle-RightWheel-nut1~Vectra-FrontAxle-RightWheel

XWHEEL~A~XWHEELNUT~A~20~nut2~XSUPERCAR~A~Vectra-FrontAxle-RightWheel-nut2~Vectra-FrontAxle-RightWheel

XWHEEL~A~XWHEELNUT~A~30~nut3~XSUPERCAR~A~Vectra-FrontAxle-RightWheel-nut3~Vectra-FrontAxle-RightWheel

XWHEEL~A~XWHEELNUT~A~40~nut4~XSUPERCAR~A~Vectra-FrontAxle-RightWheel-nut4~Vectra-FrontAxle-RightWheel

To set InContext AbsOccID in Parent Assembly only on low level components!... (This assumes the above XSUPERCAR structure exists with NO AbsOccIDs set.)

Config File

BOM FORMAT = parent_child

UPDATE BOMS = ON

CREATE BOMS = ON

BOM UPDATE OPTION = UPDATE OCCURRENCES

BOM UPDATE ID FIELDS = ChildID,AbsOccID

BOM REV RULE = RR Latest Agreed Scenery

Data File

!~ParentID~ParentRev~ChildID~ChildRev~SeqNo~AbsOccID~ContextItemID~ContextItemRevID~ContextAbsOccID

XWHEEL~A~XWHEELNUT~A~10~nut1~XSUPERCAR~A~Vectra-RearAxle-LeftWheel-nut1

XWHEEL~A~XWHEELNUT~A~20~nut2~XSUPERCAR~A~Vectra-RearAxle-LeftWheel-nut2

XWHEEL~A~XWHEELNUT~A~30~nut3~XSUPERCAR~A~Vectra-RearAxle-LeftWheel-nut3

XWHEEL~A~XWHEELNUT~A~40~nut4~XSUPERCAR~A~Vectra-RearAxle-LeftWheel-nut4

XWHEEL~A~XWHEELNUT~A~10~nut1~XSUPERCAR~A~Vectra-RearAxle-RightWheel-nut1

XWHEEL~A~XWHEELNUT~A~20~nut2~XSUPERCAR~A~Vectra-RearAxle-RightWheel-nut2

XWHEEL~A~XWHEELNUT~A~30~nut3~XSUPERCAR~A~Vectra-RearAxle-RightWheel-nut3

XWHEEL~A~XWHEELNUT~A~40~nut4~XSUPERCAR~A~Vectra-RearAxle-RightWheel-nut4

XWHEEL~A~XWHEELNUT~A~10~nut1~XSUPERCAR~A~Vectra-FrontAxle-LeftWheel-nut1

XWHEEL~A~XWHEELNUT~A~20~nut2~XSUPERCAR~A~Vectra-FrontAxle-LeftWheel-nut2

XWHEEL~A~XWHEELNUT~A~30~nut3~XSUPERCAR~A~Vectra-FrontAxle-LeftWheel-nut3

XWHEEL~A~XWHEELNUT~A~40~nut4~XSUPERCAR~A~Vectra-FrontAxle-LeftWheel-nut4

XWHEEL~A~XWHEELNUT~A~10~nut1~XSUPERCAR~A~Vectra-FrontAxle-RightWheel-nut1

XWHEEL~A~XWHEELNUT~A~20~nut2~XSUPERCAR~A~Vectra-FrontAxle-RightWheel-nut2

XWHEEL~A~XWHEELNUT~A~30~nut3~XSUPERCAR~A~Vectra-FrontAxle-RightWheel-nut3

XWHEEL~A~XWHEELNUT~A~40~nut4~XSUPERCAR~A~Vectra-FrontAxle-RightWheel-nut4

To Create an Engineering Change Item/Rev, create/attach the form type, copy the BOM child ItemRev to the problem_items pseudo folder, set Unit/EndItem effectivity when releasing the EC ItemRevision and then to create/initiate a workflow process with the EC ItemRevision as the target of the job.

*****NOTE: rerun ips_data_upload in forms mode to populate any of the created forms.***

Config File

BOM FORMAT	= parent_child
UPDATE BOMS	= ON
CREATE BOMS	= ON
BOM UPDATE OPTION	= UPDATE OCCURRENCES
BOM UPDATE ID FIELDS	= ChildID
BOM REV RULE	= Latest Working
CREATE ITEMS	= ON
CREATE REVS	= ON
UPDATE ITEMS	= ON
UPDATE REVS	= ON
SET BYPASS	= ON
CREATE DATASET	= OFF
UPDATE DATASET	= OFF
REV RELEASE RELATIONS	= IMAN_specification,BOMVIEW
CREATE STATUS	= ON
UPDATE STATUS	= LAST
CREATE EC ITEMS	= ON
UPDATE EC ITEMS	= ON
CREATE EC REVS	= ON
UPDATE EC REVS	= ON

EC TYPE = WA

ADD CHILD TO EC FOLDER = problem_items

SET EC IN BOM IC CONTEXT = ON

EC FORMS LIST TO ADD = Checklist Template Form

EC PROCESS AUTO INIT = ON

EC PROCESS TEMPLATE = CMII WA

EC PROCESS DESC = MES PROCESS NON-CONFORMANCE

Data File

!~ParentID~ParentRev~ChildID~ChildRev~ECID~ECRev~ECName~ECDesc~ECType~Status~EffEndItem~UnitStart~UnitEnd

IC_PROC1~A~IC_OP10~A~WA98765443~A~~This is a test~WA~Released~IC_PROC1~1234~1234

18. APPENDIX D: FORMS MODULE EXAMPLES

To create a "Company" form setting the form attributes "_elm_company_01" and "_elm_company_02". In this example the configuration options have all been commented out in the config file and added to the top of the date file.

Config File

#MODE = create_forms

#FORM RELATION = Reference

#FORM TYPE = Company

#FORM PARENT TYPE = ITEM

Data File

CREATE FORMS = ON

UPDATE FORMS = ON

FORM TYPE = Company

RELATION PARENT TYPE = ITEM

RELATION NAME = Reference

!~ItemID~RevID~Name~Desc~F:_elm_company_01~F:_elm_company_02

TestAssy~A~Comp FormA~Company Form~Services~Casing

Create or update items and revisions setting Project ID on the revision master form in forms mode to create a "Custom Form" form in a Specification relation, setting attributes "cust_attr_1" and "cust_attr_2".

If the revision already exists and so does the Custom Form then the Custom Form and the revision master form will be updated. Otherwise the revision and the Custom Form will be created.

Put any new items into the folder "New items".

Config File

CREATE ITEMS = ON

UPDATE ITEMS = ON

CREATE REVISIONS = ON

UPDATE REVISIONS = ON

CREATE FORMS = ON

UPDATE FORMS = ON

FORM TYPE = Custom Form

RELATION NAME = Spec

FOLDER NAME = New items

DEFAULT REV = A

Data File

ITEM TYPE = Item

!~ItemID~RevID~RM:project_id~F:cust_attr_1~F:cust_attr_2

BomAssy1~A~Project X~Value 1~Value 2

BomChild1~A~Project X~Value 1~Value 2

BomChild2~A~Project X~Value 1~Value 2

BomChild3~A~Project X~Value 1~Value 2

19. APPENDIX E: LOV MODULE EXAMPLES

To populate an LOV with string values.

Config File

LOV NAME = TEST_LOV

LOV TYPE = STRING

Data File

!~LovName~LovType~LovVal

TEST_LOV~STRING~Value 2

TEST_LOV~STRING~Value 5

TEST_LOV~STRING~Value 7

TEST_LOV~STRING~Value 100

20. APPENDIX F: RELATIONS MODULE EXAMPLES

To attach a child dataset to the parent Revision in the defined relation. The first child will be attached in a relation of Des, the second child line will move the existing child from a Requirements relation to a Specification relation and the third child will move from a reference relation to a Specification.

Config File

RELATION NAME = Ref

RELATION PARENT TYPE = REV

NEW RELATION NAME = Spec

Data File

!~ItemID~ParentRev~DsetName~DsetType~RelationName~NewRelation

#tim-00002~1~tim-00002/1~UGMASTER~Req~Spec

Engine7~A~Engine7/A~UGMASTER~~des

Engine7~O~Engine7/O~UGPART~Req~

Engine7~Q~Engine7/Q~Text~Ref~

To attach a child Item to a Parent Item Revision using the specified relation.

Config File

RELATION NAME = Ref

RELATION PARENT TYPE = REV

#NEW RELATION NAME = Spec

Data File

!~ItemID~ParentRev~ChildID~RelationName

000002~A~000003~Spec

000002~A~000003~ref

000002~A~000003~req

To attach an existing child Form to a Parent Item Revision using the specified relation.

Config File

RELATION NAME = Ref

RELATION PARENT TYPE = REV

NEW RELATION NAME = Spec

Data File

!~ItemID~ParentRev~FormName~RelationName

tim-00002~tim-00001-txt~Req

21. APPENDIX G: DATASETS MODULE EXAMPLES

The following configuration file is used in all examples shown in this section.

Config File

```
#=========================================
# General options
#=========================================
FOLDER NAME              = IMPORT_FILES
DEFAULT REV              = A
PUT FAILS IN NEWSTUFF    = ON
SET BYPASS               = ON
DEFAULT OWNER            = williamt
DEFAULT GROUP            = designer
CREATE ITEMS             = ON
UPDATE ITEMS             = ON
CREATE REVS              = ON
UPDATE REVS              = ON
UPDATE LATEST REV        = ON
#=========================================
# Dataset specific options
#=========================================
DATASET TYPE             = Text
CREATE DATASET           = ON
UPDATE DATASET           = ON
#SET OWNER BY PARENT     = OFF
#=========================================
# Dataset File Reference Options
#=========================================
FILE REF NAME            = Text
#UPDATE FILE REF NAME    = ADD REF
#UPDATE FILE REF NAME    = NEW REF
 UPDATE FILE REF NAME    = REPLACE REF
#UPDATE FILE REF NAME    = NEW VERSION
#FILE REF REPLACE OPTION        = ALL REFS
```

FILE REF REPLACE OPTION = MATCH FILE NAME

#==

Dataset Form Reference options

#==

FORM TYPE = ECRForm

UPDATE FORMS = ON

FORM REF NAME = Form

#UPDATE FORM REF NAME = ADD REF

#UPDATE FORM REF NAME = NEW REF

UPDATE FORM REF NAME = REPLACE REF

#UPDATE FORM REF NAME = NEW VERSION

#FORM REF REPLACE OPTION = MATCH FORM TYPE

FORM REF REPLACE OPTION = ALL REFS

#==

Relation details for attaching new datasets

#==

RELATION NAME = REF

RELATION PARENT TYPE = REV

Example 1: Create Item, dataset and import file

Data File

!~ItemID~RevID~Name~Desc~RM:project_id~DsetName~FileRef~File~DsetDesc~Relation Name

tim-00001~2~DS Test~Dataset Test~DS Testing~tim-00001-cfg~Text~file2_tim.cfg~A Text DS with cfg ref~Req

tim-00001~2~DS Test~Dataset Test~DS Testing~tim-00001-txt~Text~file1_tim.txt~A Text DS with txt ref

tim-00001~2~DS Test~Dataset Test~DS Testing~tim-00001-txt~Text~file2_tim.txt

Example 2: Create Item, dataset and import form and set group ownership.

Data File

!~ItemID~RevID~Name~Desc~RM:project_id~DsetName~DsetType~FormRef~FormType~F:project_id~F:series_desc~FormDesc~Group

tim-00002~A~DS Test with Form~Dataset Test with Form~DS Testing~tim-00001-txt~FormTest~Form~Item Master~DS Form Testing1~~~

tim-00002~~DS Test with Form~Dataset Test with Form~DS Testing~tim-00001-txt~FormTest~Form~CMSForm~DS Form Testing7~DS Form Testing1~DS Form Testing2~designer

Example 3: Create Item, dataset and set properties with standard system input fields (as described in Items mode).

Data File

!~ItemID~Name~Desc~RM:project_id~DsetName~FileRef~File~CrtDate~ModDate~ModBy~Owner~Group

tim-00001~DS Test~Dataset Test~DS Testing~tim-00001-txt~Text~file1_tim.txt~01-Aug-1999 00:00~02-Aug-1999~williamt~williamt~designer

tim-00001~DS Test~Dataset Test~DS Testing~tim-00001-cfg~Text~file2_tim.cfg

To create/find a dataset and Perform a Checkin on the dataset itself whilst not affecting other objects.

Config File

CREATE ITEMS = ON

UPDATE ITEMS = OFF

CREATE REVS = ON

UPDATE REVS = ON

UPDATE REV ATTACHMENTS = ON

UPDATE ITEM ATTACHMENTS = OFF

CREATE DATASET = ON

UPDATE DATASET = ON

#New CICO config options

CREATE CICO = OFF

UPDATE CICO = CHECK IN

CHECK OUT REASON = Guy's Check-Out test

CICO OBJECTS FILTER = Text

Data File

!~ItemID~RevID~DsetName~DsetType

GUY98765~A~GUYTXT_0001~Text

22. APPENDIX H: FOLDERS MODULE EXAMPLES

Creating folders.

Config File

TOP FOLDER = MasterUploadFolder
SET BYPASS = OFF
FOLDER SEPARATOR = .
TOP FOLDER DESCRIPTION = This is the top folder
DEFAULT FOLDER DESCRIPTION = This is an upload folder
PUT FAILS IN NEWSTUFF = ON
GLOBAL FODLER SEARCH = ON

Data File

FOLDER SEPARATOR = /
!~ItemID~Folder~FolderDesc
200001~TestFolder/SubFolder/SubFolder2~This is SubFolder2
200002~TestFolder3/SubFolder/SubFolder2~This is SubFolder2
200003~TestFolder4~This is TestFolder4
#000003~TestFolder/SubFolder/SubFolder3~This is SubFolder3
#000004~TestFolder/SubFolderX/SubFolder3~This is SubFolder3
#000005~TestFolder2/SubFolderY/SubFolder3~This is SubFolder3 in TestFolder2
#000006~TestFolder/SubFolderX/SubFolder3~This is SubFolder3
#000007~TestFolder/SubFolderX/SubFolder3~This is SubFolder3

23. APPENDIX I: PROJECTS MODULE EXAMPLES

Creating/Updating and Assigning Projects.

Config File

CREATE PROJECTS = ON

UPDATE PROJECTS = ON

#PROJECT ASSIGN TO TYPE = ITEM

PROJECT ASSIGN TO TYPE = REV

This option can be used to assign the latest revision when only

supplying the item ID.

UPDATE LATEST REV = ON

Use the following update options to assign to items or revisions

if not using the projects mode. If using the "projects" mode

then these are not required for the assign to work.

UPDATE ITEMS = ON

UPDATE REVS = ON

Data File

!~ProjID~ProjName~ProjDesc~ProjMemberUser~ProjMemberGroup~ProjMemberRole~ProjAdminUser~ProjPrivUser~ItemID

#PR-67890~PR-67890~Upload test project 67890~tim~Engineering~Designer~tim~tim

#PR-34567~~Upload test project 34567~tim~Engineering~Designer~tim~tim~000001

PR-34567~~Upload test project 34567~tim~Project Administration~Project Administrator~tim~tim~000001

PR-12677~~Upload test project 12677~tim~Engineering~Designer~tim~tim~000001

PR-12677~~~~~~~~000002

24. APPENDIX J: CLASSIFICATION MODULE EXAMPLES

The following configuration file is used in all examples shown in this section.

Config File

```
------------------------------
# Item and revision options...
#CREATE ITEMS = ON
#UPDATE ITEMS = ON
#CREATE REVS  = ON
#UPDATE REVS  = ON
UPDATE LATEST REV = ON

# Classification options...
#INCLASS CREATE ICO = OFF
INCLASS UPDATE ICO = OFF
#INCLASS UPDATE ICO OPTION          = SINGLE ICO ONLY
#INCLASS UPDATE ICO OPTION          = FIRST ICO
INCLASS UPDATE ICO OPTION          = ALL ICOS

#INCLASS DEFAULT CLASS ID           = 1000

INCLASS ICO CLASSIFY OBJECT        = ITEM
#INCLASS ICO CLASSIFY OBJECT        = REV

INCLASS ALLOW MULTIPLE CLASSIFICATION  = ON
# Class attributes, for reference
# Attr 1000 = Material          - String 32
# Attr 1001 = Material (LOV)    - LOV
# Attr 1002 = Length            - Real 4.2
# Attr 1003 = Diameter          - Real 4.2
```

Example 1: Create an ICO, classifying an item and updating it's ItemRevision Master form at the same time.

Data File

!~IcoPUID~IcsClassID~IcoID~ItemID~IcsAttr:1000~IcsAttr:1001~IcsAttr:1002~Ignore:IcsAttr:1003~RM:project_id

#ApNFL4wOVpZy6A~TEST-Bolts-Square~000001~000001~Some sort of metal~AL~987.5~15.87~Project Inclass

~TEST-Bolts-Square~Ico000001~000005~Some sort of metal~STL~123.5~9.1~Project Inclass

~TEST-Bolts-Square~~000005~Some sort of metal~STL~123.5~9.1~Project Inclass

~TEST-Bolts-Hex~~000005~Some sort of metal~STL~123.5~9.1~Project Inclass

#pNFL4wOVpZy6A~TEST-Bolts-Square~000001~000001~Some sort of metal~AL~987.5~15.87~Project Inclass

#~UNCLASSIFY~000001~000001~Steel~Steel~10~Project Inclass

NOTE: The ICOPuid in lines 1 and 5 was extracted using a PLMXML export. The ICOPuid allows specific ICOs to be updated. The last line was used to test unclassifying an Item.

Example 2: Creating an ICO and updating its attributes without an Item.

Data File

!~IcsClassID~IcoID~IcsAttr:1000~IcsAttr:1001~IcsAttr:1002

TEST-Bolts-Square~Ico000022~Some sort of metal~STL~1234.12

Example 3: How to delete a specific ICO give its PUID, which was taken from a PLMXML export.

Data File

!~IcoPUID~IcsClassID

AJBJZ03olhbFzC~DELETE

QFFJZwmvlhbFzC~DELETE

25. APPENDIX K: IDENTIFIERS MODULE EXAMPLES

To create an Identifier/Supplemental Identifier objects and then, on the second line, revise the Supplemental Identifier object.

Config File

CREATE ITEMS = OFF

UPDATE ITEMS = OFF

#UPDATE ITEM ATTACHMENTS = OFF

#UPDATE REV ATTACHMENTS = OFF

CREATE REVS = OFF

UPDATE REVS = OFF

SET BYPASS = ON

#UPDATE LATEST REV = ON

CREATE IDENTIFIERS = ON

UPDATE IDENTIFIERS = ON

CREATE IDFR REVS = ON

UPDATE IDFR REVS = ON

ALTERNATE ID CONTEXT = Supplier Part

Data File

!~ItemID~RevID~AlternateID~AlternateRevID~AltIDName~AltIDDesc~AltRevIDName~AltRevIDDesc~AltIDOwner~AltIDGroup

GUY987~A~BB6781~02~BB6781~A Test Altid part~PLATE 10x10~AltID revision description~bursell~dba

GUY987~B~BB6781~03~BB6781~A Test Altid part~PLATE 10x10~AltID revision description~bursell~dba

26. APPENDIX L: MANUFACTURING MODULE EXAMPLES

The following configuration file is used in the first eight examples shown in this section.

Config File

#MEProcess options
CREATE PROCESSES = ON
UPDATE PROCESSES = ON
CREATE PROCESS REVS = ON
UPDATE PROCESS REVS = ON
PROCESS TYPE = MEProcess
#MEOP options
CREATE OPERATIONS = ON
UPDATE OPERATIONS = ON
CREATE OPERATION REVS = ON
UPDATE OEPRATION REVS = ON
OPERATION TYPE = MEOP
#LINK OP TO PROCESS = OFF
LINK OP TO PROCESS = ON
#MEActivity options
CREATE ACTIVITIES = ON
UPDATE ACTIVITIES = ON
ACTIVITY TYPE = MEActivity
#MEActivity Form options
ACTIVITY FORM TYPE = RR_Activity Form
CREATE ACTIVITY FORMS = ON
UPDATE ACTIVITY FORMS = ON
#BOM module cfg options
BOM FORMAT = parent_child
UPDATE BOMS = ON
CREATE BOMS = ON
BOM UPDATE OPTION = UPDATE OCCURRENCES
#ownership
DEFAULT OWNER = bursell
DEFAULT GROUP = Trent1700

Example 1: CREATE INDEPENDENT PROCESS

Data File

ProcID~ProcRevID~ProcName~ProcType~ProcDesc

PROCESS1~A~MY PROCESS 1~MEProcess~My Process Description

Example 2: CREATE INDEPENDENT OPERATION

Data File

OpID~OpRevID~OpName~OpType~OpDesc

OPN1~A~MY OPERATION 1~MEOP~My Operation Description

Example 3: CREATE/UPDATE OPERATION AND CREATE/UPDATE ACTIVITY ON OPERATION

Data File

OpID~OpRevID~ActName~ActDesc~ActType~ActTime

OPN2~A~ACT1~My Activity Description~MEActivity~43.2

Example 4: CREATE/UPDATE OPERATION AND CREATE/UPDATE ACTIVITY ON OPERATION & CREATE/UPDATE ACTIVITY FORM + ATTRIBUTES

Data File

OpID~OpRevID~ActName~ActDesc~ActType~ActTime~FormName~FormType~FormDesc ~F:rr_activity_detail_text~F:rr_activity_std_text

OPN2~A~ACT1~My Activity Description~MEActivity~43.2~ACT1FORM~RR_Activity Form~My Form Description~the first attribute value~the,second,attr,array,values

Example 5: CREATE/UPDATE PROCESS+OPS and BUILD BOM (Process<--Operation). This requires LINK OP TO PROCESS = ON configuration option is set.

Data File

ProcID~ProcRevID~OpID~OpRevID~SeqNo

PROCESS1~A~OPN1~A~10

PROCESS1~A~OPN2~A~20

Example 6: ADD MULTIPLE ACTIVITIES TO AN OPERATION

Data File

OpID~OpRevID~ActName~ActDesc~ActType~ActTime~FormName~FormType~FormDesc~F:rr_activity_detail_text~F:rr_activity_std_text

OPN2~A~ACT1~My Activity Description~MEActivity~43.2~ACT1FORM~RR_Activity Form~My Form Description~the first attribute value~the,second,attr,array,values

OPN2~A~ACT2~My Activity Description 2~MEActivity~5~ACT2FORM~RR_Activity Form~My Form Description~the first attribute value~the,second,attr,array,values

Example 7: ADD ADD ACTIVITY SEQUENCING (FLOWS)

Data File

ProcID~ProcRevID~OpID~OpRevID~SeqNo~Predecessor~ActName~ActPreds

PROC1~A~OP1~A~10~ACT1~

PROC1~A~OP1~A~20~ACT2~ACT1

Example 8: ADD OPERATION SEQUENCING (FLOWS)

Data File

ProcID~ProcRevID~OpID~OpRevID~SeqNo~Predecessor

PROC1~A~OP1~A~10~

PROC1~A~OP2~A~20~OP1

To consume occurrences from XSUPERCAR structure (See sample in BOMs Mode) with InContext Absolute Occurrence data set. Create Process structure in the same file.

Config File

```
------------------------------
#MEProcess options
CREATE PROCESSES        = ON
UPDATE PROCESSES        = ON
CREATE PROCESS REVS     = ON
UPDATE PROCESS REVS     = ON
PROCESS TYPE            = MEProcess
#MEOP options
CREATE OPERATIONS       = ON
UPDATE OPERATIONS       = ON
CREATE OPERATION REVS   = ON
UPDATE OEPRATION REVS   = ON
OPERATION TYPE          = MEOP
LINK OP TO PROCESS      = ON
#MEActivity options
CREATE ACTIVITIES       = ON
UPDATE ACTIVITIES       = ON
ACTIVITY TYPE           = MEActivity
#MEActivity Form options
ACTIVITY FORM TYPE      = RR_Activity Form
CREATE ACTIVITY FORMS   = ON
UPDATE ACTIVITY FORMS   = ON
#MEWorkArea options
CREATE WORKAREAS        = ON
UPDATE WORKAREAS        = ON
CREATE WORKAREA REVS    = ON
UPDATE WORKAREA REVS    = ON
WORKAREA TYPE           = MEWorkarea
#BOM module cfg options (Mandatory)
BOM FORMAT              = parent_child
UPDATE BOMS             = ON
CREATE BOMS             = ON
BOM UPDATE OPTION       = UPDATE OCCURRENCES
BOM UPDATE ID FIELDS    = ChildID,AbsOccID
```

BOM REV RULE = RR Latest Agreed Scenery

#Ownership

DEFAULT OWNER = bursell

DEFAULT GROUP = Trent1700

LINK PRODUCT TO PROCESS = ON

LINK PLANT TO PROCESS = ON

Data File

!~ProcID~ProcRevID~ProcType~ProcName~ProcDesc~OpID~OpRevID~OpType~OpNam
e~OpDesc~SeqNo~Predecessor~ActName~ActType~ActTime~ActDesc~ActPreds~Produc
tTargetID~ProductTargetRevID~PlantTargetID~PlantTargetRevID~OccType~TargetChildID
~TargetChildRevID~TargetChildAbsOccID~TargetChildSeqNo~ProcChildID~~~

GPROC1~A~MEProcess~GPROC1-NAME~Process
Description~GOP1~A~MEOP~OP1~Op1 Description~10~~10~MEActivity~23.76~Activity
10~~~~~~~~~~~~~~

GPROC1~A~~~~GOP1~A~MEOP~OP1~Op1 Description~10~~20~MEActivity~99~Activity
20~10~~~~~~~~~~~~~

GPROC1~A~~~~GOP1~A~MEOP~OP1~Op1 Description~10~~30~MEActivity~10~Activity
30~20~~~~~~~~~~~~~

GPROC1~A~~~~GOP2~A~MEOP~OP2~Op2
Description~20~GOP1~~~~~~~~~~~~~~~~~~

GPROC1~A~~~~GOP3~A~MEOP~OP3~Op3
Description~30~GOP2~~~~~~~~~~~~~~~~~~

GPROC1~A~~~~GOP4~A~MEOP~OP4~Op4
Description~40~GOP3~~~~~~~~~~~~~~~~~~

GPROC1~A~~~~GOP4~A~MEOP~OP4~~40~~10~MEActivity~123.909~Activity
10~~~~~~~~~~~~~~

GPROC1~A~~~~GOP4~A~MEOP~OP4~~40~~20~MEActivity~14~Activity
20~10~~~~~~~~~~~~~

GPROC1~A~~~~~~~~~~~~~~~~XSUPERCAR~A~~~MEConsumed~XWHEEL~A~Vectra-
RearAxle-LeftWheel~~GOP1~~~

GPROC1~A~~~~~~~~~~~~~~~~XSUPERCAR~A~~~MEConsumed~XWHEEL~A~Vectra-
RearAxle-RightWheel~~GOP2~~~

GPROC1~A~~~~~~~~~~~~~~~~XSUPERCAR~A~~~MEConsumed~XWHEEL~A~Vectra-
FrontAxle-LeftWheel~~GOP3~~~

GPROC1~A~~~~~~~~~~~~~~~~XSUPERCAR~A~~~MEConsumed~XWHEEL~A~Vectra-
FrontAxle-RightWheel~~GOP4~~~

GPROC1~A~~~~~~~~~~~~~~~~XSUPERCAR~A~~~MEConsumed~XWHEEL~A~Vectra-
FrontAxle-RightWheel~~GOP4~~~

GPROC1~A~~~~~~~~~~~~~~~~~~STWORK1~A~MEWorkArea~STWORK0001~A~F1_W
orkArea1~~GOP1~~~

To Release an MEOP Revision, and also Status any MEActivities (any other other attachments to Activity_Root/MEActivities).

Config File

CREATE ITEMS = OFF

UPDATE ITEMS = OFF

CREATE REVS = OFF

UPDATE REVS= ON

REV RELEASE RELATIONS = SPECIFICATION,BOMVIEW,ACTIVITIES

UPDATE REVS ATTACHS STATUS = ON

UNRELEASE REVS = ON

CREATE STATUS = ON

#UPDATE STATUS = UNRELEASE

Data File

!~ItemID~RevID~Status

GOP1~A~Released

TCG02000105315~A~Released