THROUGH LIFE SUPPORT STANDARD

Guidance on writing DEXlib TLSS Templates

LSC REFERENCE: ECSMODTLSS5025.120

ESL REFERENCE: ESUKPC09.000157

Authors	Checked	Approved
Brad Harris - Eurostep Mike Ward - Eurostep	Rob Bodington - Eurostep Tim Turner - LSC Group	Simon Dick - Eurostep Paul Clark - LSC Group
Issue	**Date**	**Client**
1.0	01 November 2007	TLSD Policy Coordination - TLSS Project Manager

LSC Group Ltd.
Lincoln House
Fradley Park
Lichfield
Staffordshire
WS13 8RZ
United Kingdom

Eurostep Limited,
Cwttir Lane,
St. Asaph,
Denbighshire,
LL17 0LQ,
United Kingdom

Amendment Record

Issue	Date	Summary of changes
1.0	2007-11-01	First Issue, delivered by TLSS Work Package 1, Sub-task 1

Distribution

This document has been distributed to:

Copy No	Media	Location	Title
1	PDF File		TLSS Project Manager

TABLE OF CONTENTS

1 INTRODUCTION

1. In accordance with the TLSS DEX Development Methodology document [1], a TLSS Business Template[1] shall be created for each TLSS Business Object[2].
2. It is assumed that the business object, for which the template is being created, has already been fully defined and reviewed according to [2].
3. Usually the TLSS requirement described by a business object will straightforwardly result in a single template.
4. When there is optionality (i.e. optional or zero cardinality or selects or subtypes) in a TLSS business object this will often be dealt with by using characterization or classification in the resulting business template. Sometimes, however, the only way to deal with optionality in a business object will be to create more than one template – one template for each option which the TLSS community may wish to instantiate. In such cases, the relevant business object UML diagram should be produced in different edited versions that reflect the different options (see, for example, the contrasting UML diagrams in business templates: "actual_part" and "batch").
5. As stated in [1], if it is possible to define a generic PLCS template that fully meets the requirements of the business object, then this is the preferred option, rather than making the template specific to TLSS. Even if this is not the case, it may be possible to define a generic PLCS template that satisfies a large proportion of the TLSS requirement, and then to define an additional TLSS business template that specializes it – perhaps with additional characterizations and / or reference data. In this way, maximum re-use can be made of both PLCS and TLSS work.
6. Guidance on writing PLCS Capability based templates can be seen at http://www.plcs-resources.org/plcs/dexlib/help/dex/dvlp_cap.htm. This includes a description of the contents of templates.
7. It is assumed that the accounts and software listed below are activated and set up / installed.

2 KNOWLEDGE, SKILLS AND TOOLS REQUIRED

1. The process requires a thorough understanding of:
   - The TLSS process model and the information flow classes that it defines.
   - UML information modelling methods and practices.
   - The TLSS business object model.
   - The existing TLSS business templates.[3]
   - The existing PLCS templates.
   - The PLCS data model.
   - The template instantiation path language.
   - Xml and how files are controlled by DTD's.
2. The following accounts and software are required:
   - DEXLib developer account on Sourceforge. See DEXlib help file http://www.plcs-resources.org/plcs/dexlib/help/dex/dvlp_intro.htm.
   - CVS and associated encryption software (Putty, Plink, Puttygen, Pageant) – for downloading and uploading files to Sourceforge. See DEXlib help file http://www.plcs-resources.org/plcs/dexlib/help/dex/dexlib_cvs_access.htm

[1] Here after referred to as "the template"

[2] Here after referred to as "the business object"

[3] Note that the existing templates need to be updated iaw the new business template structure.

- DEXlib – including utilities for creating the DEX and its constituent files. See DEXlib help file http://www.plcs-resources.org/plcs/dexlib/help/dex/dexlib_cvs_access.htm. Note that this tool needs two parameters setting (Using Options->Set DEXLib properties). These are the path to the DEXLib root (such as "C:\Sourceforge\dexlib") and the path to the eep executable (such as C:\Sourceforge\dexlib\utils\dex).
- Xml editor such as Oxygen (http://www.oxygenxml.com/) or XMLSpy (http://www.altova.com/products/xmlspy/xml_editor.html) – for editing the template.xml file. The uploading of invalid xml to the DEXlib Sourceforge site can cause significant maintenance problems.
- MSVisio – if Graphical Express and the DEX Template plug-in are used.
- Graphical Express and the associated DEX Template – for creating template model diagrams (see http://www.plcs-resources.org/plcs/dexlib/help/dex/sw_graphexp.htm) - or drawing software that can produce similar results.
- Graphical Instance – for creating template instance diagrams (see http://www.plcs-resources.org/plcs/dexlib/help/dex/sw_graphinst.htm) – or drawing software that can produce similar results.

NOTE: The examples used in the following sections are all derived from the TLSS business template "actual_part".

3 GUIDANCE NOTES ON CREATION OF A TLSS BUSINESS TEMPLATE

3.1 Add your contact details

1. Add your details to the contacts file: "\dexlib\data\basic\dex\contacts.xml" if they have not yet been added.[4]
2. The entry for the information model diagram in the contacts.xml file should follow the example below:

```
<contact id="bradharris">
        <firstname>Brad</firstname>
        <lastname>Harris</lastname>
        <affiliation>PLCS Inc / Eurostep Limited</affiliation>
        <street>Cwttir Lane</street>
        <city>St Asaph</city>
        <state>Denbighshire</state>
        <postcode>LL17 0LQ</postcode>
        <country>UK</country>
        <phone> +44 7798 674520</phone>
        <email>brad.harris@eurostep.com</email>
</contact>
```

3. Upload this file to Sourceforge:
    a. Add your access key using the Pageant utility.

[4] Note: the "\" character is used for the folder separator in path names defined in this document – on the assumption that Windows will be the Operating System used by DEXLib developers.

b. Run wincvs.
c. In the left hand upper window, navigate to "\dexlib\data\basic\dex\".
d. The "contacts.xml" file should be marked with a red page at the left hand side of its name in the right hand upper window.
e. Right click on the file name and select "Commit" from the drop down menu.
f. Enter a log message in the window that appears and then hit the "OK" button.
g. When the Commit is complete, the red page at the left hand side of the file name should change back to white and the Rev number (second property column) should increment.

3.2 Create the Business Template

1. To create a TLSS Business Template run the DEXlib Tool and go to *Tools>Create Business template*:

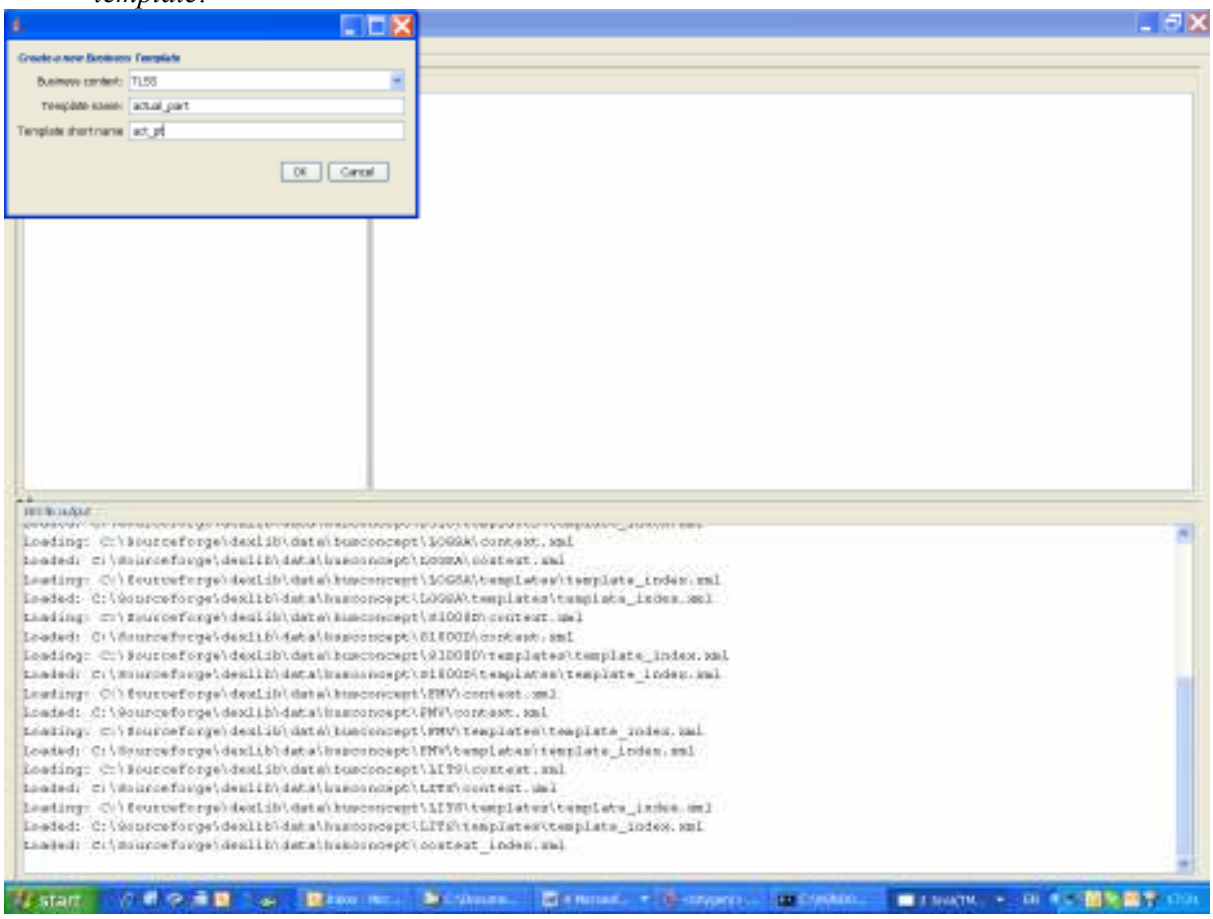

Figure 1 Screen shot of the DEXLib Tool User Interface for Creation of a TLSS Business Template

2. The *Business Context* drop down box shall be edited to select "TLSS".
3. The *Template name* field shall be edited with the identification of the Template. This shall be the same as the name given to the business object for which it is being defined. This identification string is used as the DEXLib folder name, under which all the template files will be stored, as well as the *name* element within the main template file – "template.xml". Note that the name given to all template files that will be uploaded to Sourceforge shall be in a lower_case_underscore_separated style. NO CAPITALS and no white space are allowed in any cvs file names.

4. The *Template short name* field shall be edited with a short name – *act_part* – in this case. These short names may be necessary when the template is used on diagrams of other templates – if there are graphical layout restrictions – but in general they shall not be used further by other TLSS templates and DEXs.
5. The template *name* and *short_name* shall be unique within the TLSS *business_context*.
6. On completion of the three fields and hitting the "OK" button, the following message appears:

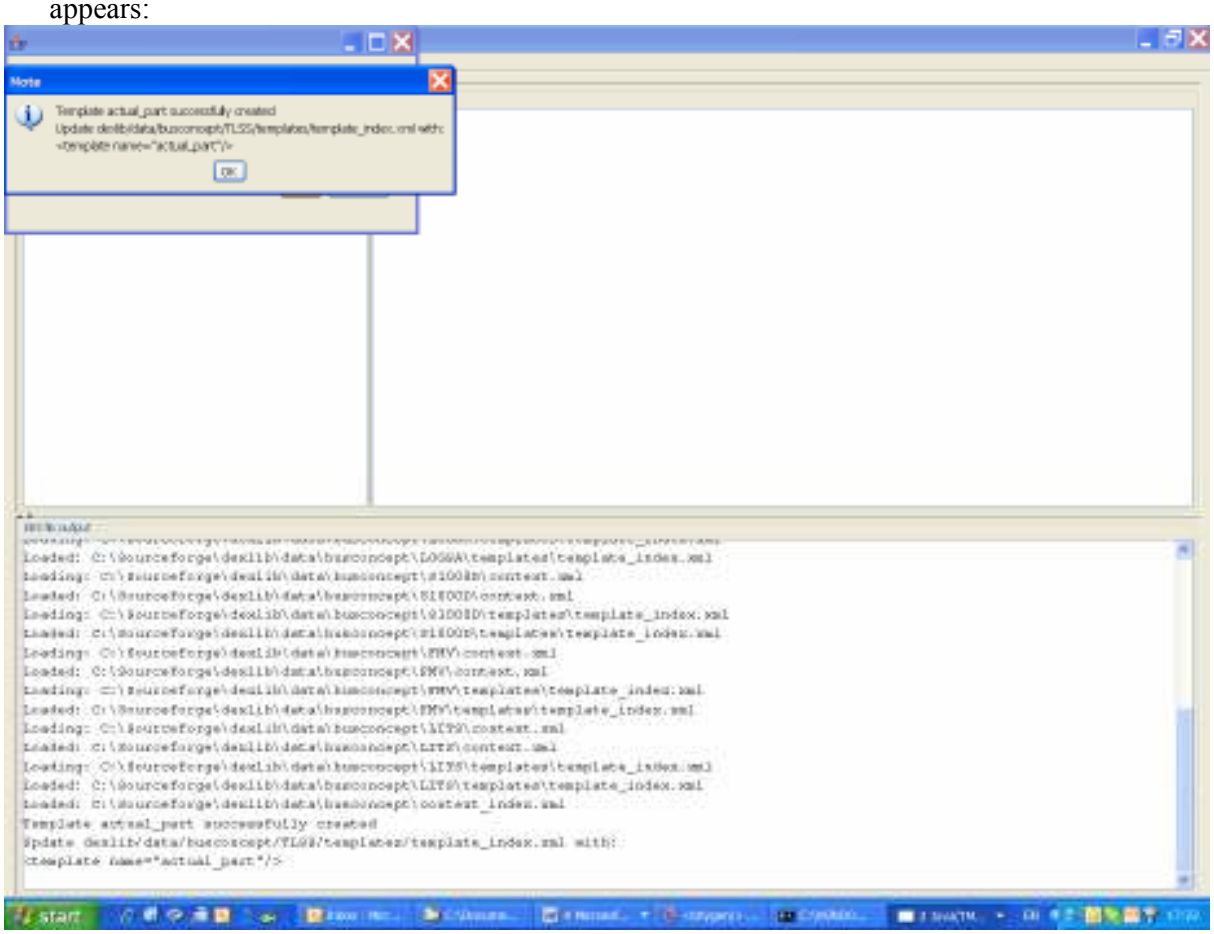

Figure 2 Screen shot of the DEXLib Tool User Interface on Completion of Creation of a TLSS Business Template

This confirms that the template file structure has been created (under "dexlib\data\busconcept\TLSS\templates" with folder name "actual_part" – in this case).

7. The DEXLib Tool will also create three subfolders under "dexlib\data\busconcept\TLSS\templates\actual_part":
   - "\dvlp":
     a. A skeleton file called "issues.xml" is generated by the script. This can be edited to record issues against the template.
     b. Source diagrams for information models shall be stored in here as well, typically in graphic file formats produced by UML tools.
   - "\images":
     a. Graphic files for inclusion in DEXlib shall be converted to png format and stored in this directory.
   - "\sys":

a. The files that are generated in this folder are used by the DEXlib and CVS software packages for management purposes and shall not be edited manually.

NOTE: No alterations shall be made to the directory structure generated by the DEXLib Tool and, apart from template.xml, none of the files that are created in "dexlib\data\busconcept\TLSS\templates\actual_part" shall be edited manually.

3.3 Edit the index file

1. The message above also prompts the user to update the appropriate index file by adding the identification of the new template to it. The index file "template_index.xml" is located in "dexlib\data\busconcept\TLSS\templates". For each TLSS business template created, a line has to be added to the *<templates>* element of this file. This element requires the *name (*as entered above).
2. The entry for the information model diagram in the "template_index.xml" file should follow the example below:

   <template name="actual_part"/>

3. The lines within the *<templates>* element shall be ordered alphabetically by name, since the sequential ordering of the elements within the *<templates>* element is used as the display order of the TLSS templates within the browser view of DEXLib.

3.4 Edit the template files

1. The main file that needs to be edited by the developer is called "template.xml". This contains both "header" type elements as well as the descriptive, model and specification elements, as described below.
2. In addition to these elements, the Reference Data that constitutes part of the detailed technical specification of the template is referenced via the input parameter definitions and the instantiation path – see sections [3.4.7] and [3.4.10]. The creation of TLSS Reference Data is described in separate TLSS guidance documentation [3] and [4].
3. Note that if any part of the "template.xml" file is created by copying text from other applications such as MS Word, care should be taken to remove certain characters such as smart quotation marks. It is better to create text in a text editor. Other characters that require care are the ampersand, less-than, and greater-than (& < >), which are characters that have a special interpretation in xml. These may be escaped - if it is essential to use such characters – as follows: & < >.

3.4.1 Header elements

1. The *name, short_name,* and *business_context* elements are populated by the DEXLib Tool from the user input when the template is created.[5]
2. There are three elements (*rcs.author*, *rcs.date* and *rcs.revision*) at the head of the file that are populated by the CVS software when the file is first checked into Sourceforge and these shall not be manually edited.
3. The *development.folder* element is populated by the DEXLib Tool when the DEX is created and shall not be manually edited.
4. The remainder of the file must be edited by the developer.

[5] Note: if there is a business_concept="" element in the file, this shall be ignored. Future releases of the DEXLib Tool will not generate this element.

5. The next element is the *status* element and this requires *state* and *completion_date* attributes. The valid values for *state* are defined in the file "dexlib\dtd\dex\dexdoc.ent". The value "in_work" is suggested as the initial *state* for all TLSS DEXs. The *completion_date* shall be taken from the TLSS project plan under which the DEX is being developed.
6. The entry for status in the "template.xml" file should follow the example below:

```
<status state="in_work" completion_date="2007-12-30">
        <review type="model" reviewer="mikeward" status="not_started"
        completion_date=""/>
        <review type="business" reviewer="andy.burden" status="not_started"
        completion_date=""/>
</status>
```

Note that the *status* element is an optional component of a "template.xml" file, but for TLSS templates it shall be populated and updated according to the progress of the DEX.

3.4.2 Contacts

1. An *editor* contact must be provided for the template, in addition to those named in the status element.
2. The entry for contacts in the "template.xml" file should follow the example below:

```
<contacts>
        <editor ref="bradharris"/>
</contacts>
```

3. Additional reviewer contacts may also be added:

```
<contacts>
        <editor ref="bradharris"/>
        <reviewer ref="mikeward" type="model"/>
        <reviewer ref="andy.burden" type="business"/>
</contacts>
```

Note that the names referenced must also be present within the "dexlib\data\basic\dex\contacts.xml" file.

3.4.3 Description

1. The *description* element of the business template shall contain a brief overview of its purpose and the information that is being mapped / represented in PLCS terms by the template.
2. The entry for description in the "template.xml" file should follow the example below:

```
<description>
        <p>
                This template describes how to represent the TLSS concept of an actual part
                in terms of PLCS model elements (templates, entities, and reference data).
        </p>
</description>
```

3.4.4 Business Perspective

1. The <*business_perspective*> element of the template shall describe the TLSS data exchange specifications of which it is a part. A general description of typical exchanges, with reference back to relevant TLSS DEX specifications, is sufficient to define the business perspective.
2. This element is not generated by the DEXLib Tool since it is optional, but for all TLSS business templates it shall be included manually by the user and completed as illustrated below.
3. The entry for business perspective in the "template.xml" file should follow the example below:

<*business_perspective*>
<*p*>
The ActualPart business object is used by those TLSS Data Exchange Specifications that require information about
actual parts / realized parts / items of equipment / assets that are identified by an individual serial number.
</*p*>
<*p*>
Usually, ActualPart business objects are originally created by the manufacturer of the item and are an external input to TLSS processes. Such business objects may be updated by TLSS processes if, for example, UID markings are added in-service.
</*p*>
</*business_perspective*>

4. A business perspective element may contain many types of element allowed in DEXLib text sections, including paragraphs, sections, figures, tables, lists, and references to other DEXLib items.

3.4.5 Business object definition (Information requirement)

1. The <*business_object_definition*> element shall describe the business object for which the template is created.
2. This element is not generated by the DEXLib Tool since it is optional, but for all TLSS business templates it shall be included manually by the user and completed as illustrated below.
3. The element shall include:
   - a copy of the MagicDraw definition of the business object for which the template is created. See [2].
   - A UML composite structure diagram of the business object, showing all attributes and relationships.
   - A table containing all attribute and relationship names and definitions, together with their relationship to their corresponding TLSS information requirement.
4. A *business_object_definition* element may contain many types of element allowed in DEXLib text sections, including paragraphs, sections, figures, tables, lists, and references to other DEXLib items. For the purpose of TLSS, this element shall contain the elements described below.

3.4.5.1 Business object definition: introductory paragraph

1. The <p> element shall describe the business object for which the template is created.

2. The entry for the introductory paragraph in the “template.xml” file should follow the example below:

```
<business_object_definition>
        <p>
                The definition of the actual_part object is:
                This is data about an actual part, which is a physical instance of a
                manufacturers item (design).
        </p>
</business_object_definition>
```

3.4.5.2 Business object definition: UML diagram reference

1. The < *figure*> element shall contain a link to the png file containing the UML diagram for the business object.
2. The entry for the UML diagram reference in the “template.xml” file should follow the example below:

```
<business_object_definition>
        <figure number="1" id="actual_part_xg">
                <title>Actual part</title>
                <img src="images/actual_part.png"/>
        </figure>
</business_object_definition>
```

3.4.5.3 Business object definition: UML diagram

1. The png file containing the UML diagram for the business object shall be placed in the “dexlib\data\busconcept\TLSS\templates*$template_name*\images” directory and named in accordance with the src attribute of the img element above.
2. The UML diagram in the “$template_name.png” file should follow the example below:

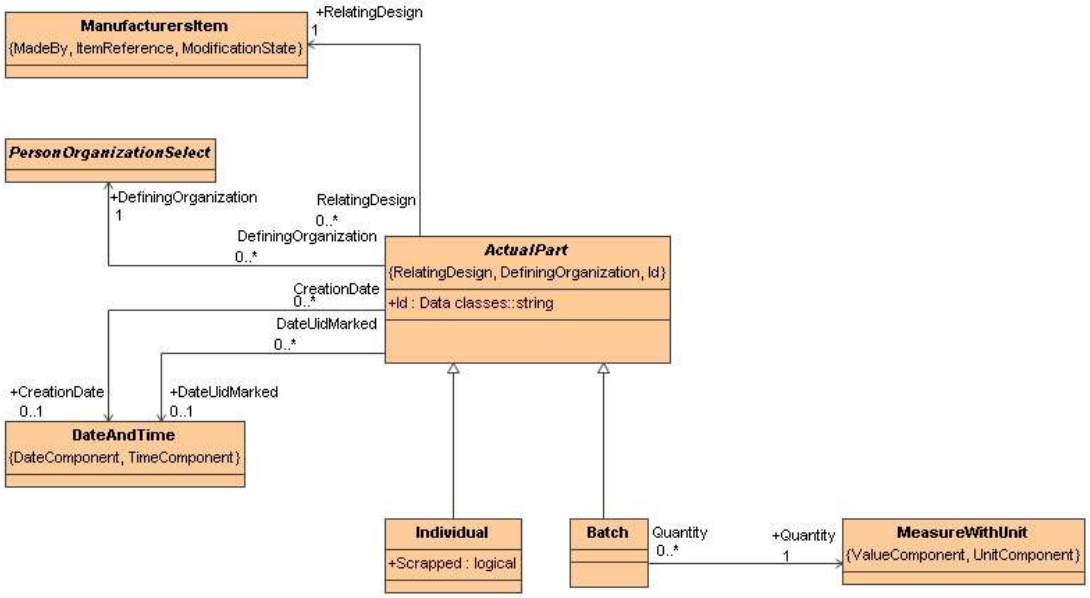

Figure 3 Graphical Representation for Business Object ActualPart[6]

3.4.5.4 Business object definition: attribute details table

1. The detail for each TLSS attribute shall be presented in the form of a table as shown below:

Business Object	*Attribute*	*Definition*	*Type*	*Information Requirement*
ActualPart	***N/A***	***This is data about an actual part, which is a physical instance of a manufacturers item (design).***	***N/A***	***Actual part***
	Id	*This is the identifier of the actual part as assigned by the defining organization (often a manufacturers serial number). In some circumstances / applications, the actual part owner / operator / maintainer will allocate asset identifiers to actual parts for the purpose of asset tracking and management and / or job scheduling. If both manufacturers serial number and additional asset identifiers are required to be managed, then alias identifiers should be used to create the latter.*	*string*	*Actual part.id*
	DefiningOrganization	*This is the reference to the person / organisation that assigned the id (and probably manufactured the actual part).*	*PersonOrganizationSelect*	*Actual part.defining_organization*
	RelatingDesign	*This is the reference to the manufacturers item (design) of which the actual part is a physical instance.*	*ManufacturersItem*	*Actual part.relating_design*
	CreationDate	*This is the date and time on which the actual part was created..*	*DateAndTime*	*Actual part.creation_date*
	DateUidMarked	*This is the date and time on which the UID was associated with or marked on the actual part.*	*DateAndTime*	*N/A*[7]

[6] Batch is not represented in the actual_part template.

Individual	***N/A***	***This is data about an individual instance of a manufacturers item that has been produced.***	***N/A***	***Individual***
	Scrapped	*This is an indicator that states whether the actual part has been scrapped.*	*logical*	*Individual.scrapped*

Figure 4 Table of Definitions for the ActualPart Business Object

3.4.6 Information model diagram

1. The *<model_diagrams>* element is where the information model diagram, which is the means by which the set of PLCS entities and templates that will represent the business object, is defined.
2. The GraphicalExpress tool and the associated DEX Template tool or similar software shall be used for drawing these diagrams. Instructions on how to run these tools are detailed in the help files under the EXPRESS-G > GraphicalEXPRESS Help menu.
3. This is the point in the DEX development process when the need for consistent mapping from source requirements / business objects to the PLCS data model is absolutely essential. Mapping guidelines are defined in [5]. The template developer needs to have a thorough understanding of the semantics of both the business object and the whole of the PLCS data model, as well as the PLCS templates that have already been developed. They also need a good understanding of the use of GraphicalExpress and the associated DEX Template.
4. The focal point of the diagram is likely to be a single PLCS template or entity. For example, when defining the TLSS actual_part business template, the corresponding PLCS concept is product_as_realized, and therefore the focal point for the mapping is the representing_product_as_realized template.
5. Once the main mapping point has been decided, the mandatory attributes and relationships of the business object need to be mapped to corresponding PLCS entities and templates that can be associated with the main entity / template. For example, the TLSS ActualPart maps to a PLCS Product_as_individual.
6. If no mapping can be defined to the ISO 10303-239 (PLCS) data model, and MoD requires such a mapping, then use should be made of the ISO TC 184, SC4 Standards Enhancement and Discrepancy System (SEDS) process outlined in [6].
7. Naming of template occurrences and parameters on model diagrams shall be in accordance with the source requirement / business object names wherever possible. This is the only means of recording a mapping from requirement to solution that can be documented for traceability.
8. All reference parameters provided by underlying OASIS templates should be shown on the model diagram. Additional EXPRESS Entities that form part of the template and which may be referenced by other TLSS templates should also be given reference parameter names and shown on the diagram at this stage[8]. Reference parameters that are unused by or renamed in the current business template will be indicated in the model_diagram_example below.
9. When mapping questions or issues arise, these shall be discussed with other TLSS template developers whereby a common understanding and agreement can be reached. Input can also be sought from other template developers through the plcs-dex@lists.oasis-open.org email exploder. If agreement can not be reached, then the MOD TLSS Project Manager (or nominated deputy) shall have the final decision.

[7] Needs to be added.

[8] The Product_group_membership entity in the actual_part template is not required as a reference parameter.

3.4.6.1 Information model diagram: EXPRESSG/Template diagram reference

1. The <*model_diagram*> element shall contain a description and a link to the png file containing the combined EXPRESS-G and template diagram for the business template.
2. The master attribute of the <*img*> element shall be set to match the name of the file in which the figure was originally created (in the example a GraphicalExpress/Visio file) and this file should be placed in the "dexlib\data\busconcept\TLSS\templates*$template_name*\dvlp" directory
3. The entry for the EXPRESS-G and template diagram reference in the "template.xml" file should follow the example below:

```
<model_diagrams>
        <model_diagram diagram_type="EXPRESS-G">
                <description>
                        The graphical representation of actual_part template.
                </description>
                <figure id="actual_part_tmpl">
                        <title>The graphical representation of the actual_part
                        template</title>
                        <img src="images/actual_part_tmpl.png"
                        master="dvlp/actual_part_tmpl.vsd"/>
                </figure>
        <model_diagrams>
<model_diagram diagram_type="EXPRESS-G">
```

3.4.6.2 Information model diagram: EXPRESSG/Template diagram

1. The png file containing the combined EXPRESS-G and template diagram for the business template shall be placed in the "dexlib\data\busconcept\TLSS\templates*$template_name*\images" directory and named in accordance with the src attribute of the <*img*> element above.
2. The Information model diagram in the "$template_name_tmpl.png" file should follow the example below:

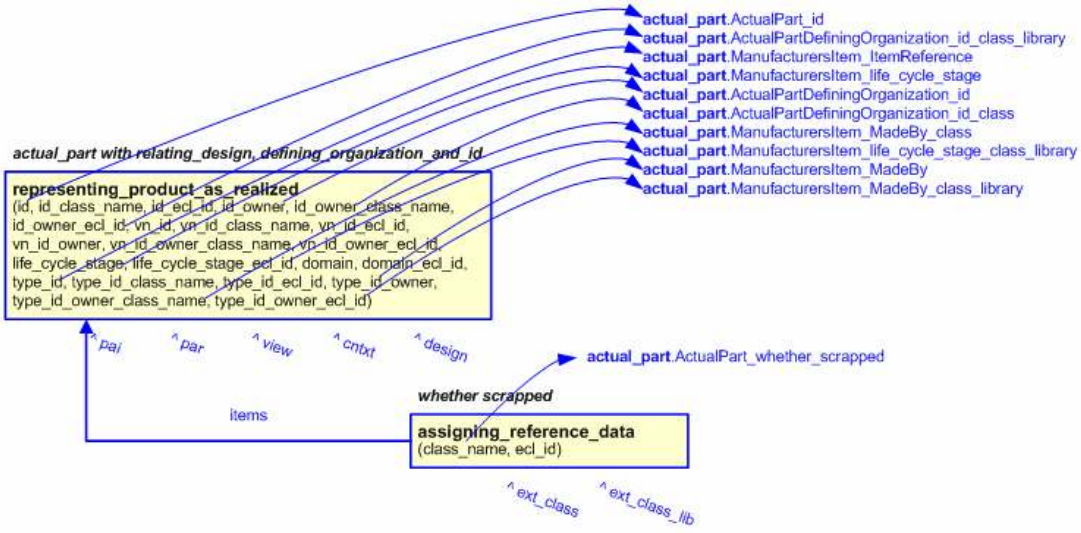

Figure 5 Information model diagram for the actual_part Business Template

NOTE: optional attributes and relationships (such as CreationDate and DateUidMarked) shall be defined as "Characterizations" (below) and not shown on the model diagram.

3.4.7 Input parameters

1. Template input parameters are the names of the PLCS attributes that correspond to the business object attributes; plus zero or more additional attributes that do not appear in the business object.
2. Any additional attributes in a list of business template input parameters will be of three types: (1) attributes corresponding to additional mandatory information in the OASIS template which the TLSS community decide to populate (such as the ActualPart_life_cycle_stage input parameter[9]); (2) attributes which provide additional semantics to an attribute that does correspond directly to a business object attribute (such as the ActualPartDefiningOrganization_id_class parameter which indicates whether a particular identifier for a company is a simple name, or a CAGE code, or some other type of identifier); and (3) attributes corresponding to relationships between EXPRESS datatypes that are created in the new business template (not applicable in this example).
3. Any input parameters of type (1) or (2) which the TLSS community does not wish to use explicitly can be set to "/NULL" or to a default value (such as "urn:plcs:rdl:tlss") in the instantiation path and can, thereby, be pruned from the explicit list of input parameters.
4. All TLSS business template input parameters should be given names that correspond to business concepts with which the TLSS community are familiar and should be named with full, meaningful names, constructed from the concatenation of business object names, attribute names, additional terms, and underscores. The use of non-standard abbreviations is deprecated on the grounds of intelligibility. This may result in long parameter names, but it will enable direct traceability to the source attributes of each parameter.
5. This is the point in the DEX development process where the need for TLSS reference data becomes apparent, since it is usually necessary to add semantics to the generic parameters by classifying them in terms derived from the business object they represent. For example, an "identification_assignment" to an "actual_part" template is classified as an "actual_part_identifier". These can be specified as part of the input parameter definition and hyperlinks will then be generated to those reference data items that have already been defined.
6. Any reference data items required by TLSS should be added to the "dexlib\data\refdata\ plcs-rdl-tlss.owl" file using Protégé.
7. The entry for the input parameters in the "template.xml file" should follow the example below:

```
<params_in>

        <!-- actual part id -->
        <param_in name="ActualPart_id" type="STRING">
                <description>
                        The identifier or serial number of the actual part.
                </description>
        </param_in>
```

[9] Which is used in this example to represent the "scrapped" attribute of ActualPart.

```
<!-- actual part defining organization -->
<param_in name="ActualPartDefiningOrganization_id" type="STRING">
        <description>
                The name or identifier of the organization that defines the actual part
                identifier.
        </description>
</param_in>

<param_in name="ActualPartDefiningOrganization_id_class" type="CLASS"
default="Organization_name">
        <classifications>
                <class id="Organization_identification_code"
                urn="urn:plcs:rdl:std"/>
                <class id="Organization_name" urn="urn:plcs:rdl:std"/>
        </classifications>
        <description>
                The name of the class used to classify the DefiningOrganization
                identifier. For example CAGE code, or organization name.
        </description>
</param_in>

<param_in name="ActualPartDefiningOrganization_id_class_library"
type="URN" default="urn:plcs:rdl:std">
        <description>
                The id of the <express_ref
                linkend="external_class:arm:External_class_arm.External_class_lib
                rary"/>
                 storing the definition of the class referenced by the parameter
                @ActualPartDefiningOrganization_id_class class.
                 This parameter should be set to the default.
        </description>
</param_in>

<param_in name="ActualPart_whether_scrapped" type="CLASS">
        <classifications>
                <class id="Scrapped_actual_part" urn="urn:plcs:rdl:tlss"/>
                <class id="Servicable_actual_part" urn="urn:plcs:rdl:tlss"/>
        </classifications>
        <description>
                A classification of the actual part that indicates whether or not the
                part has been scrapped.
        </description>
</param_in>


<!-- manufacturer's item identitifer -->
<param_in name="ManufacturersItem_ItemReference" type="STRING">
        <description>
                The identifier of the design of the actual_part.
        </description>
```

```
</param_in>

<param_in name="ManufacturersItem_MadeBy" type="STRING">
        <description>
                The name or identifier of the organization responsible for the
                Manufacturer's item.
        </description>
</param_in>

<param_in name="ManufacturersItem_MadeBy_class" type="CLASS">
        <classifications>
                <class id="Organization_identification_code"
                urn="urn:plcs:rdl:std"/>
                <class id="Organization_name" urn="urn:plcs:rdl:std"/>
        </classifications>
        <description>
                The name of the class used to classify the
                 identification of the organization responsible for the Manufacturer's
                item. For example CAGE code, or organization name.
        </description>
</param_in>

<param_in name="ManufacturersItem_MadeBy_class_library" type="URN"
default="urn:plcs:rdl:std">
        <description>
                The id of the <express_ref
                linkend="external_class:arm:External_class_arm.External_class_lib
                rary"/>
                 storing the definition of the class referenced by the parameter
                @ManufacturersItem_MadeBy_class.
                 This parameter should be set to the default.
        </description>
</param_in>

<!-- actual part life cycle stage (not included in ActualPart section of TLSS but
should be populated with an appropriate classification) -->
<param_in name="ManufacturersItem_life_cycle_stage" type="CLASS">
        <classifications>
        <!-- a list of all possible classifications that can be used -->
        <class id="Development_stage" urn="urn:plcs:rdl:std"/>
        <class id="Production_stage" urn="urn:plcs:rdl:std"/>
        <class id="Retirement_stage" urn="urn:plcs:rdl:std"/>
        <class id="Support_stage" urn="urn:plcs:rdl:std"/>
        <class id="Utilization_stage" urn="urn:plcs:rdl:std"/>
        </classifications>
        <description>
                A classification of the life cycle stage of the actual part.
        </description>
</param_in>
```

```
<param_in name="ManufacturersItem_life_cycle_stage_class_library"
type="URN" default="urn:plcs:rdl:std">
    <description>
        The identifier of the
        <express_ref
        linkend="external_class:arm:External_class_arm.External_class_lib
        rary"/>
        storing the definition of the class referenced by the parameter
        @ManufacturersItem_life_cycle_stage.
    </description>
</param_in>
</params_in>
```

8. These parameters are based on those for the PLCS template used here (representing_product_as_realized).

3.4.8 Reference parameters

1. Template reference parameters are names for EXPRESS datatypes that can be referenced when the template is referred to in the instantiation path of another template.
2. All TLSS business template reference parameters shall be named to reflect the business object elements to which they correspond. Again, the use of abbreviations is discouraged.
3. Reference parameters inherited from the underlying OASIS template can be pruned at this stage (if they are not relevant in a TLSS context) or renamed.
4. The names of the reference parameters used in the PLCS template (representing_product_as_realized) and the replacement reference parameter names used in the TLSS actual_part template are listed below:

PLCS entity	PLCS template	PLCS ref param name	TLSS ref param name
Product_as_individual	representing_product_as_realized	^pai	^actual_part
Product_as_realized	representing_product_as_realized	^par	n/a
Product_as_individual_view	representing_product_as_realized	^view	n/a
View_definition_context	representing_product_as_realized	^cntxt	n/a
Part	representing_product_as_realized	^design	^manufacturers_item

5. The two of these that are relevant in the TLSS context for actual_part are documented in the "template.xml" file as follows:

```
<params_ref>
  <!-- The reference parameters -->
  <param_ref name="actual_part" entity_type="Product_as_individual"/>
  <param_ref name="manufacturers_item" entity_type="Part"/>
</params_ref>
```

3.4.9 Model diagram example

1. The <model_diagram_example> element may be populated once the input parameters and reference parameters have been defined.
2. The GraphicalExpress tool and the associated DEX Template tool can automatically generate the required consolidated template diagram which includes all input parameters and reference parameters defined for the business template and which can be reused in other business templates.

3.4.9.1 Model diagram example: Consolidated template diagram reference

1. The < *model_diagram_example*> element shall contain a description and a link to the png file containing the consolidated template diagram for the business template.
2. The master attribute of the <*img*> element shall be set to match the name of the file in which the figure was originally created (in the example a GraphicalExpress/Visio file) and this file should be placed in the "dexlib\data\busconcept\TLSS\templates*$template_name*\dvlp" directory.

3. The entry for the consolidated template diagram reference in the "template.xml" file should follow the example below:

```
<model_diagrams>
        <model_diagram_example diagram_type="EXPRESS-G">
                <description>
                        The graphical representation of actual_part template.
                </description>
                <figure id="actual_part_tmpl">
                        <title>The graphical representation of the actual_part
                        template</title>
                        <img src="images/actual_part_tmpl.png"
                        master="dvlp/actual_part_tmpl.vsd"/>
                </figure>
         </model_diagram_example>
</model_diagrams>
```

3.4.9.2 Model diagram example: Consolidated template diagram

1. The png file containing the consolidated template diagram for the business template for the business template shall be placed in the "dexlib\data\busconcept\TLSS\templates*$template_name*\images" directory and named in accordance with the src attribute of the <*img*> element above.
2. The consolidated template diagram in the "$template_name_tmpl.png" file should follow the example below:

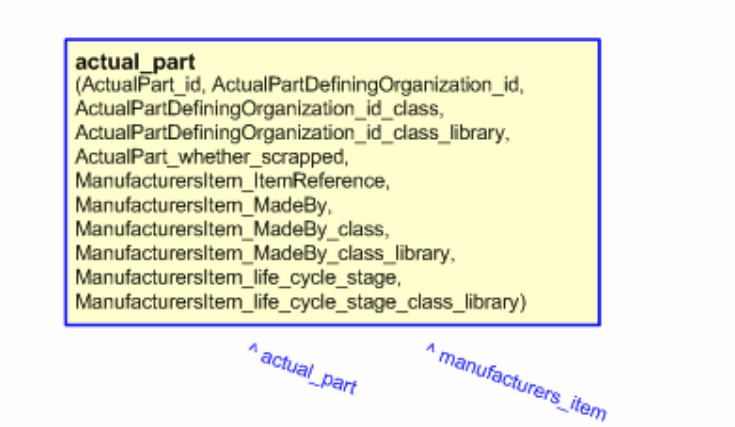

Figure 6 Model diagram example for the actual_part Business Template

3.4.10 Instantiation path

1. The instantiation path is the means of formally specifying the set of PLCS EXPRESS datatypes that must be populated in order to represent the business object for which the template is created; the data (or range of data) that must be used to populated each datatype; and the mapping between the input parameters and the underlying EXPRESS datatypes. Instantiation path syntax is documented in DEXLib > info pages > toc > templates >Template instantiation path.
2. The entry for the instantiation_path in the "template.xml" file should follow the example below:

```
-- instantiate ActualPart
    /representing_product_as_realized(id=@ActualPart_id,
    id_class_name='Actual_part_identifier',
    id_ecl_id='urn:plcs:rdl:tlss',
    id_owner=@ActualPartDefiningOrganization_id,
    id_owner_class_name=@ActualPartDefiningOrganization_id_class,
    id_owner_ecl_id=@ActualPartDefiningOrganization_id_class_library,
    vn_id='/NULL',
    vn_id_class_name='Product_as_realized_identification_code',
    vn_id_ecl_id='urn:plcs:rdl:std',
    vn_id_owner='/NULL',
    vn_id_owner_class_name='Organization_name',
    vn_id_owner_ecl_id='urn:plcs:rdl:std',
    life_cycle_stage=@ManufacturersItem_life_cycle_stage,
    life_cycle_stage_ecl_id=@ManufacturersItem_life_cycle_stage_class_library,
    domain='Through_life_support_standard',
    domain_ecl_id='urn:plcs:rdl:tlss',
    type_id=@ManufacturersItem_ItemReference,
    type_id_class_name='Manufacturers_item_reference',
    type_id_ecl_id='urn:plcs:rdl:tlss',
    type_id_owner=@ManufacturersItem_MadeBy,
    type_id_owner_class_name=@ManufacturersItem_MadeBy_class,
    type_id_owner_ecl_id=@ManufacturersItem_MadeBy_class_library)/

-- assign ref parameters
    %^actual_part = $representing_product_as_realized.pai%
    %^manufacturers_item = $representing_product_as_realized.par%

-- assign whether_scrapped classification
    /assigning_reference_data(items=^actual_part,
    class_name=@ActualPart_whether_scrapped,
    ecl_id='urn:plcs:rdl:tlss')/
```

3.4.11 Template example

1. The <*template_example*> element is populated with a series of example values for the input parameters listed earlier in the file. This populated example serves as the basis of the instantiation diagrams presented below.
2. The entry for the <*template_example*> in the "template.xml" file should follow the example below:

```
<template_example name="actual_part">
    <param_in name="ActualPart_id" value="#1"/>
    <param_in name="ActualPartDefiningOrganization_id" value="ACME Engineering"/>
    <param_in name="ActualPartDefiningOrganization_id_class"
    value="Organization_name"/>
    <param_in name="ActualPartDefiningOrganization_id_class_library"
    value="urn:plcs:rdl:std"/>
    <param_in name="ActualPart_whether_scrapped" value="Scrapped_actual_part"/>
    <param_in name="ManufacturersItem_ItemReference" value="#2"/>
    <param_in name="ManufacturersItem_MadeBy" value="ACME Design"/>
    <param_in name="ManufacturersItem_MadeBy_class" value="Organization_name"/>
    <param_in name="ManufacturersItem_MadeBy_class_library" value="urn:plcs:rdl:std"/>
    <param_in name="ManufacturersItem_life_cycle_stage" value="Support_stage"/>
    <param_in name="ManufacturersItem_stage_class_library" value="urn:plcs:rdl:std"/>
</template_example>
```

3.4.12 Unique rules

1. This is a list of input parameters that must be assigned a unique value within the context of an exchange file.
2. This section is optional within the “template_xml” file.
3. Unique rules are determined by the business object specification.

Any entry for the <*unique_rules*>

4. in the “template.xml” file should follow the example below:

```
<unique_rules>
    <unique_rule_param name="ActualPart_ur">
    <unique_param param="ActualPart_id"/>
    <unique_param param="ActualPartDefiningOrganization_id"/>
    <unique_param param="ManufacturersItem_ItemReference"/>
    <unique_entity ref_param_in_path="actual_part"/>
    </unique_rule_param>

    <unique_rule_param name="ManufacturersItem_ur">
    <unique_param param="ManufacturersItem_ItemReference"/>
    <unique_param param="ManufacturersItem_MadeBy"/>
    <unique_param param="ManufacturersItem_life_cycle_stage"/>
    <unique_entity ref_param_in_path="manufacturers_item"/>
    </unique_rule_param>
</unique_rules>
```

3.4.13 Instance diagrams

1. The <instance_diagrams> element may be populated once the template example and the instantiation path have been defined.
2. The GraphicalInstance tool may be used to generate suitable instance diagrams. Instructions for the use of this tool can be found under “help”.

3.4.13.1 Instance diagrams: Instance diagram reference

1. The <*instance_diagram_instantiation*> element shall contain a template_example_proxy (which inserts an incarnation of the <*template_example*> described above); a link to the png

file containing the instance diagram for the business template; and a <*exchange_file*> element.

2. The master attribute of the <*img*> element shall be set to match the name of the file in which the figure was originally created (in the example a GraphicalInstance file) and this file should be placed in the "dexlib\data\busconcept\TLSS\templates*$template_name*\dvlp" directory
3. The content of the <*exchange_file*> element can be generated automatically using GraphicalInstance or constructed manually using a text editor and this file should be placed in the "dexlib\data\busconcept\TLSS\templates*$template_name*\dvlp" directory with the name "*$template_name*.p21". The contents of this file may then be copied to the <*exchange_file*> element.
4. The entry for the instance diagram instantiation in the "template.xml" file should follow the example below:

```
<instance_diagrams>
  <instance_diagram_instantiation diagram_type="EXPRESS-I">
    <template_example_proxy name="actual_part"/>
    <figure id="actual_part_inst">
          <title>Entities instantiated by actual_part template</title>
          <img src="images/actual_part_inst.png" master="dvlp/actual_part_inst.giz"/>
    </figure>
    <exchange_file type="p21">
        #287 = EXTERNAL_CLASS($,$,$,$);
        #286 = CLASSIFICATION_ASSIGNMENT(#287,(#279),'/IGNORE');
        #284 = EXTERNAL_CLASS_LIBRARY('default',$);
        #283 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#284);
        #282 = EXTERNAL_CLASS($,$,$,$);
        #281 = CLASSIFICATION_ASSIGNMENT(#282,(#279),'/IGNORE');
        #279 = VIEW_DEFINITION_CONTEXT('/IGNORE','/IGNORE','/IGNORE');
        #278 = PART_VIEW_DEFINITION('/IGNORE','/IGNORE','/IGNORE',#279,(),#255);
        #277 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
        #276 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#277);
        #275 = EXTERNAL_CLASS($,$,$,$);
        #274 = CLASSIFICATION_ASSIGNMENT(#275,(#272),'/IGNORE');
        #272 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#264,'/IGNORE',(#
257));
        #15 = CLASSIFICATION_ASSIGNMENT(#16,(#13),'/IGNORE');
        #13 = IDENTIFICATION_ASSIGNMENT('ACME
Engineering','/IGNORE','/IGNORE',(#11));
        #11 = ORGANIZATION('/IGNORE','/IGNORE');
        #9 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:tlss',$);
        #8 = EXTERNAL_CLASS('/NULL','Actual_part_identifier','/IGNORE',#9);
        #7 = EXTERNAL_CLASS($,$,$,$);
        #6 = CLASSIFICATION_ASSIGNMENT(#7,(#4),'/IGNORE');
        #4 = IDENTIFICATION_ASSIGNMENT('1','/IGNORE',$,(#2));
        #2 = PRODUCT_AS_INDIVIDUAL('/IGNORE','/IGNORE','/IGNORE');
        #303 = EXTERNAL_CLASS($,$,$,$);
        #302 = CLASSIFICATION_ASSIGNMENT(#303,(#300),'/IGNORE');
        #300 = IDENTIFICATION_ASSIGNMENT('1','/IGNORE',$,(#298));
```

```
#298 = PRODUCT_AS_INDIVIDUAL('/IGNORE','/IGNORE','/IGNORE');
#295 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:tlss',$);
#294 = EXTERNAL_CLASS('/NULL','Scrapped_actual_part','/IGNORE',#295);
#293 = EXTERNAL_CLASS($,$,$,$);
#292 = CLASSIFICATION_ASSIGNMENT(#293,(#171),'/IGNORE');
#290 = PRODUCT_DESIGN_TO_INDIVIDUAL(#230,#171);
#289 = EXTERNAL_CLASS_LIBRARY('default',$);
#288 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#289);
#31 = EXTERNAL_CLASS('/NULL','Version_identification_code','/IGNORE',#32);
#30 = EXTERNAL_CLASS($,$,$,$);
#29 = CLASSIFICATION_ASSIGNMENT(#30,(#27),'/IGNORE');
#27 = IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE',$,(#25));
#25 = PRODUCT_AS_REALIZED('/IGNORE','/IGNORE',#2);
#24 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#23 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#24);
#22 = EXTERNAL_CLASS($,$,$,$);
#21 = CLASSIFICATION_ASSIGNMENT(#22,(#19),'/IGNORE');
#19 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#11,'/IGNORE',(#4)
);
#18 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#17 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#18);
#16 = EXTERNAL_CLASS($,$,$,$);
#319 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#320);
#318 = EXTERNAL_CLASS($,$,$,$);
#317 = CLASSIFICATION_ASSIGNMENT(#318,(#315),'/IGNORE');
#315 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#307,'/IGNORE',(#
300));
#314 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#313 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#314);
#312 = EXTERNAL_CLASS($,$,$,$);
#311 = CLASSIFICATION_ASSIGNMENT(#312,(#309),'/IGNORE');
#309 = IDENTIFICATION_ASSIGNMENT('ACME
Engineering','/IGNORE','/IGNORE',(#307));
#307 = ORGANIZATION('/IGNORE','/IGNORE');
#305 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:tlss',$);
#304 = EXTERNAL_CLASS('/NULL','Actual_part_identifier','/IGNORE',#305);
#47 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#46 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#47);
#45 = EXTERNAL_CLASS($,$,$,$);
#44 = CLASSIFICATION_ASSIGNMENT(#45,(#42),'/IGNORE');
#42 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#34,'/IGNORE',(#2
7));
#41 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#40 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#41);
#39 = EXTERNAL_CLASS($,$,$,$);
#38 = CLASSIFICATION_ASSIGNMENT(#39,(#36),'/IGNORE');
#36 = IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE','/IGNORE',(#34));
```

```
#34 = ORGANIZATION('/IGNORE','/IGNORE');
#32 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#335 = EXTERNAL_CLASS($,$,$,$);
#334 = CLASSIFICATION_ASSIGNMENT(#335,(#332),'/IGNORE');
#332 = IDENTIFICATION_ASSIGNMENT('/NULL','/IGNORE','/IGNORE',(#330));
#330 = ORGANIZATION('/IGNORE','/IGNORE');
#328 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#327 =
EXTERNAL_CLASS('/NULL','Product_as_realized_identification_code','/IGNORE',#328);
#326 = EXTERNAL_CLASS($,$,$,$);
#325 = CLASSIFICATION_ASSIGNMENT(#326,(#323),'/IGNORE');
#323 = IDENTIFICATION_ASSIGNMENT('/NULL','/IGNORE',$,(#321));
#321 = PRODUCT_AS_REALIZED('/IGNORE','/IGNORE',#298);
#320 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#63 = EXTERNAL_CLASS($,$,$,$);
#62 = CLASSIFICATION_ASSIGNMENT(#63,(#60),'/IGNORE');
#60 = IDENTIFICATION_ASSIGNMENT('ACME
Engineering','/IGNORE','/IGNORE',(#58));
#58 = ORGANIZATION('/IGNORE','/IGNORE');
#56 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#55 = EXTERNAL_CLASS('/NULL','Serial_identification_code','/IGNORE',#56);
#54 = EXTERNAL_CLASS($,$,$,$);
#53 = CLASSIFICATION_ASSIGNMENT(#54,(#51),'/IGNORE');
#51 = IDENTIFICATION_ASSIGNMENT('1','/IGNORE',$,(#49));
#49 = PRODUCT_AS_INDIVIDUAL('/IGNORE','/IGNORE','/IGNORE');
#350 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#349 = EXTERNAL_CLASS('/NULL','Support_stage','/IGNORE',#350);
#348 = EXTERNAL_CLASS($,$,$,$);
#347 = CLASSIFICATION_ASSIGNMENT(#348,(#345),'/IGNORE');
#345 = VIEW_DEFINITION_CONTEXT('/IGNORE','/IGNORE','/IGNORE');
#344 =
PRODUCT_AS_INDIVIDUAL_VIEW('/IGNORE','/IGNORE','/IGNORE',#345,(),#321);
#343 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#342 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#343);
#341 = EXTERNAL_CLASS($,$,$,$);
#340 = CLASSIFICATION_ASSIGNMENT(#341,(#338),'/IGNORE');
#338 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#330,'/IGNORE',(#
323));
#337 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#336 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#337);
#79 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#78 =
EXTERNAL_CLASS('/NULL','Product_as_individual_identification_code','/IGNORE',#79);
#77 = EXTERNAL_CLASS($,$,$,$);
#76 = CLASSIFICATION_ASSIGNMENT(#77,(#74),'/IGNORE');
#74 = IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE',$,(#72));
#72 = PRODUCT_AS_REALIZED('/IGNORE','/IGNORE',#49);
#71 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#70 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#71);
```

```
#69 = EXTERNAL_CLASS($,$,$,$);
#68 = CLASSIFICATION_ASSIGNMENT(#69,(#66),'/IGNORE');
#66 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#58,'/IGNORE',(#5
1));
#65 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#64 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#65);
#366 = ORGANIZATION('/IGNORE','/IGNORE');
#364 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:tlss',$);
#363 =
EXTERNAL_CLASS('/NULL','Manufacturers_item_reference','/IGNORE',#364);
#362 = EXTERNAL_CLASS($,$,$,$);
#361 = CLASSIFICATION_ASSIGNMENT(#362,(#359),'/IGNORE');
#359 = IDENTIFICATION_ASSIGNMENT('2','/IGNORE',$,(#357));
#357 = PART('/IGNORE','/IGNORE','/IGNORE');
#355 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:tlss',$);
#354 =
EXTERNAL_CLASS('/NULL','Through_life_support_standard','/IGNORE',#355);
#353 = EXTERNAL_CLASS($,$,$,$);
#352 = CLASSIFICATION_ASSIGNMENT(#353,(#345),'/IGNORE');
#95 =
PRODUCT_AS_INDIVIDUAL_VIEW('/IGNORE','/IGNORE','/IGNORE',#96,(),#72);
#94 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#93 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#94);
#92 = EXTERNAL_CLASS($,$,$,$);
#91 = CLASSIFICATION_ASSIGNMENT(#92,(#89),'/IGNORE');
#89 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#81,'/IGNORE',(#7
4));
#88 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#87 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#88);
#86 = EXTERNAL_CLASS($,$,$,$);
#85 = CLASSIFICATION_ASSIGNMENT(#86,(#83),'/IGNORE');
#83 = IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE','/IGNORE',(#81));
#81 = ORGANIZATION('/IGNORE','/IGNORE');
#382 = PART_VERSION('/IGNORE','/IGNORE',#357);
#381 = PRODUCT_CATEGORY('/IGNORE','part','/IGNORE');
#380 = PRODUCT_CATEGORY_ASSIGNMENT(#381,(#357));
#379 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#378 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#379);
#377 = EXTERNAL_CLASS($,$,$,$);
#376 = CLASSIFICATION_ASSIGNMENT(#377,(#374),'/IGNORE');
#374 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#366,'/IGNORE',(#
359));
#373 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#372 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#373);
#371 = EXTERNAL_CLASS($,$,$,$);
#370 = CLASSIFICATION_ASSIGNMENT(#371,(#368),'/IGNORE');
```

```
#368 = IDENTIFICATION_ASSIGNMENT('ACME
Design','/IGNORE','/IGNORE',(#366));
#110 = IDENTIFICATION_ASSIGNMENT('2','/IGNORE',$,(#108));
#108 = PART('/IGNORE','/IGNORE','/IGNORE');
#106 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#105 = EXTERNAL_CLASS('/NULL','Product_life_cycle_support','/IGNORE',#106);
#104 = EXTERNAL_CLASS($,$,$,$);
#103 = CLASSIFICATION_ASSIGNMENT(#104,(#96),'/IGNORE');
#101 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#100 = EXTERNAL_CLASS('/NULL','Support_stage','/IGNORE',#101);
#99 = EXTERNAL_CLASS($,$,$,$);
#98 = CLASSIFICATION_ASSIGNMENT(#99,(#96),'/IGNORE');
#96 = VIEW_DEFINITION_CONTEXT('/IGNORE','/IGNORE','/IGNORE');
#399 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#391,'/IGNORE',(#
384));
#398 = EXTERNAL_CLASS_LIBRARY('default',$);
#397 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#398);
#396 = EXTERNAL_CLASS($,$,$,$);
#395 = CLASSIFICATION_ASSIGNMENT(#396,(#393),'/IGNORE');
#393 =
IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE','/IGNORE',(#391));
#391 = ORGANIZATION('/IGNORE','/IGNORE');
#389 = EXTERNAL_CLASS_LIBRARY('default',$);
#388 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#389);
#387 = EXTERNAL_CLASS($,$,$,$);
#386 = CLASSIFICATION_ASSIGNMENT(#387,(#384),'/IGNORE');
#384 = IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE',$,(#382));
#127 = CLASSIFICATION_ASSIGNMENT(#128,(#125),'/IGNORE');
#125 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#117,'/IGNORE',(#
110));
#124 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#123 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#124);
#122 = EXTERNAL_CLASS($,$,$,$);
#121 = CLASSIFICATION_ASSIGNMENT(#122,(#119),'/IGNORE');
#119 =
IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE','/IGNORE',(#117));
#117 = ORGANIZATION('/IGNORE','/IGNORE');
#115 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:tlss',$);
#114 =
EXTERNAL_CLASS('/NULL','Manufacturers_item_reference','/IGNORE',#115);
#113 = EXTERNAL_CLASS($,$,$,$);
#112 = CLASSIFICATION_ASSIGNMENT(#113,(#110),'/IGNORE');
#415 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#416);
#414 = EXTERNAL_CLASS($,$,$,$);
#413 = CLASSIFICATION_ASSIGNMENT(#414,(#406),'/IGNORE');
#411 = EXTERNAL_CLASS_LIBRARY('default',$);
#410 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#411);
#409 = EXTERNAL_CLASS($,$,$,$);
```

```
#408 = CLASSIFICATION_ASSIGNMENT(#409,(#406),'/IGNORE');
#406 = VIEW_DEFINITION_CONTEXT('/IGNORE','/IGNORE','/IGNORE');
#405 = PART_VIEW_DEFINITION('/IGNORE','/IGNORE','/IGNORE',#406,(),#382);
#404 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#403 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#404);
#402 = EXTERNAL_CLASS($,$,$,$);
#401 = CLASSIFICATION_ASSIGNMENT(#402,(#399),'/IGNORE');
#142 = ORGANIZATION('/IGNORE','/IGNORE');
#140 = EXTERNAL_CLASS_LIBRARY('default',$);
#139 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#140);
#138 = EXTERNAL_CLASS($,$,$,$);
#137 = CLASSIFICATION_ASSIGNMENT(#138,(#135),'/IGNORE');
#135 = IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE',$,(#133));
#133 = PART_VERSION('/IGNORE','/IGNORE',#108);
#132 = PRODUCT_CATEGORY('/IGNORE','part','/IGNORE');
#131 = PRODUCT_CATEGORY_ASSIGNMENT(#132,(#108));
#130 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#129 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#130);
#128 = EXTERNAL_CLASS($,$,$,$);
#422 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:tlss',$);
#421 = EXTERNAL_CLASS('/NULL','Scrapped_actual_part','/IGNORE',#422);
#420 = EXTERNAL_CLASS($,$,$,$);
#419 = CLASSIFICATION_ASSIGNMENT(#420,(#298),'/IGNORE');
#417 = PRODUCT_DESIGN_TO_INDIVIDUAL(#357,#298);
#416 = EXTERNAL_CLASS_LIBRARY('default',$);
#159 = CLASSIFICATION_ASSIGNMENT(#160,(#157),'/IGNORE');
#157 = VIEW_DEFINITION_CONTEXT('/IGNORE','/IGNORE','/IGNORE');
#156 = PART_VIEW_DEFINITION('/IGNORE','/IGNORE','/IGNORE',#157,(),#133);
#155 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#154 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#155);
#153 = EXTERNAL_CLASS($,$,$,$);
#152 = CLASSIFICATION_ASSIGNMENT(#153,(#150),'/IGNORE');
#150 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#142,'/IGNORE',(#
135));
#149 = EXTERNAL_CLASS_LIBRARY('default',$);
#148 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#149);
#147 = EXTERNAL_CLASS($,$,$,$);
#146 = CLASSIFICATION_ASSIGNMENT(#147,(#144),'/IGNORE');
#144 =
IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE','/IGNORE',(#142));
#175 = CLASSIFICATION_ASSIGNMENT(#176,(#173),'/IGNORE');
#173 = IDENTIFICATION_ASSIGNMENT('1','/IGNORE',$,(#171));
#171 = PRODUCT_AS_INDIVIDUAL('/IGNORE','/IGNORE','/IGNORE');
#168 = PRODUCT_DESIGN_TO_INDIVIDUAL(#108,#49);
#167 = EXTERNAL_CLASS_LIBRARY('default',$);
#166 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#167);
#165 = EXTERNAL_CLASS($,$,$,$);
#164 = CLASSIFICATION_ASSIGNMENT(#165,(#157),'/IGNORE');
#162 = EXTERNAL_CLASS_LIBRARY('default',$);
```

```
#161 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#162);
#160 = EXTERNAL_CLASS($,$,$,$);
#191 = EXTERNAL_CLASS($,$,$,$);
#190 = CLASSIFICATION_ASSIGNMENT(#191,(#188),'/IGNORE');
#188 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#180,'/IGNORE',(#
173));
#187 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#186 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#187);
#185 = EXTERNAL_CLASS($,$,$,$);
#184 = CLASSIFICATION_ASSIGNMENT(#185,(#182),'/IGNORE');
#182 = IDENTIFICATION_ASSIGNMENT('ACME
Engineering','/IGNORE','/IGNORE',(#180));
#180 = ORGANIZATION('/IGNORE','/IGNORE');
#178 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:tlss',$);
#177 = EXTERNAL_CLASS('/NULL','Actual_part_identifier','/IGNORE',#178);
#176 = EXTERNAL_CLASS($,$,$,$);
#207 = CLASSIFICATION_ASSIGNMENT(#208,(#205),'/IGNORE');
#205 = IDENTIFICATION_ASSIGNMENT('/NULL','/IGNORE','/IGNORE',(#203));
#203 = ORGANIZATION('/IGNORE','/IGNORE');
#201 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#200 =
EXTERNAL_CLASS('/NULL','Product_as_realized_identification_code','/IGNORE',#201);
#199 = EXTERNAL_CLASS($,$,$,$);
#198 = CLASSIFICATION_ASSIGNMENT(#199,(#196),'/IGNORE');
#196 = IDENTIFICATION_ASSIGNMENT('/NULL','/IGNORE',$,(#194));
#194 = PRODUCT_AS_REALIZED('/IGNORE','/IGNORE',#171);
#193 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#192 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#193);
#223 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#222 = EXTERNAL_CLASS('/NULL','Support_stage','/IGNORE',#223);
#221 = EXTERNAL_CLASS($,$,$,$);
#220 = CLASSIFICATION_ASSIGNMENT(#221,(#218),'/IGNORE');
#218 = VIEW_DEFINITION_CONTEXT('/IGNORE','/IGNORE','/IGNORE');
#217 =
PRODUCT_AS_INDIVIDUAL_VIEW('/IGNORE','/IGNORE','/IGNORE',#218,(),#194);
#216 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#215 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#216);
#214 = EXTERNAL_CLASS($,$,$,$);
#213 = CLASSIFICATION_ASSIGNMENT(#214,(#211),'/IGNORE');
#211 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#203,'/IGNORE',(#
196));
#210 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
#209 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#210);
#208 = EXTERNAL_CLASS($,$,$,$);
#239 = ORGANIZATION('/IGNORE','/IGNORE');
#237 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:tlss',$);
#236 =
EXTERNAL_CLASS('/NULL','Manufacturers_item_reference','/IGNORE',#237);
```

```
        #235 = EXTERNAL_CLASS($,$,$,$);
        #234 = CLASSIFICATION_ASSIGNMENT(#235,(#232),'/IGNORE');
        #232 = IDENTIFICATION_ASSIGNMENT('2','/IGNORE',$,(#230));
        #230 = PART('/IGNORE','/IGNORE','/IGNORE');
        #228 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:tlss',$);
        #227 =
EXTERNAL_CLASS('/NULL','Through_life_support_standard','/IGNORE',#228);
        #226 = EXTERNAL_CLASS($,$,$,$);
        #225 = CLASSIFICATION_ASSIGNMENT(#226,(#218),'/IGNORE');
        #255 = PART_VERSION('/IGNORE','/IGNORE',#230);
        #254 = PRODUCT_CATEGORY('/IGNORE','part','/IGNORE');
        #253 = PRODUCT_CATEGORY_ASSIGNMENT(#254,(#230));
        #252 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
        #251 = EXTERNAL_CLASS('/NULL','Owner_of','/IGNORE',#252);
        #250 = EXTERNAL_CLASS($,$,$,$);
        #249 = CLASSIFICATION_ASSIGNMENT(#250,(#247),'/IGNORE');
        #247 =
ORGANIZATION_OR_PERSON_IN_ORGANIZATION_ASSIGNMENT(#239,'/IGNORE',(#
232));
        #246 = EXTERNAL_CLASS_LIBRARY('urn:plcs:rdl:std',$);
        #245 = EXTERNAL_CLASS('/NULL','Organization_name','/IGNORE',#246);
        #244 = EXTERNAL_CLASS($,$,$,$);
        #243 = CLASSIFICATION_ASSIGNMENT(#244,(#241),'/IGNORE');
        #241 = IDENTIFICATION_ASSIGNMENT('ACME
Design','/IGNORE','/IGNORE',(#239));
        #271 = EXTERNAL_CLASS_LIBRARY('default',$);
        #270 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#271);
        #269 = EXTERNAL_CLASS($,$,$,$);
        #268 = CLASSIFICATION_ASSIGNMENT(#269,(#266),'/IGNORE');
        #266 =
IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE','/IGNORE',(#264));
        #264 = ORGANIZATION('/IGNORE','/IGNORE');
        #262 = EXTERNAL_CLASS_LIBRARY('default',$);
        #261 = EXTERNAL_CLASS('/NULL','default','/IGNORE',#262);
        #260 = EXTERNAL_CLASS($,$,$,$);
        #259 = CLASSIFICATION_ASSIGNMENT(#260,(#257),'/IGNORE');
        #257 = IDENTIFICATION_ASSIGNMENT('Unknown','/IGNORE',$,(#255));
    </exchange_file>
  </instance_diagram_instantiation>
</instance_diagrams>
```

3.4.13.2 Instance diagrams: Instance diagram

1. The png file containing the instance diagram for the business template shall be placed in the "dexlib\data\busconcept\TLSS\templates*$template_name*\images" directory and named in accordance with the src attribute of the <*img*> element above.
2. The instance diagram in the "$template_name_inst.png" file should follow the example below:

Figure 7 Instance diagram example for the actual_part Business Template

This template instance diagram is the content of the instance_diagram_instantiation element and is named actual_part_inst.png. It represents a populated example of the entities and templates that compose the business template.

3.4.13.3 Instance diagrams: Instance example diagram reference

1. The *<instance_diagram_example>* element shall contain a template_example_proxy (which inserts an incarnation of the *<template_example>* described above) and a link to the png file containing the instance example diagram for the business template.
2. The master attribute of the *<img>* element shall be set to match the name of the file in which the figure was originally created (in the example a GraphicalInstance file) and this file should be placed in the "dexlib\data\busconcept\TLSS\templates*$template_name*\dvlp" directory
3. The entry for the instance example diagram instantiation in the "template.xml" file should follow the example below:

```
<instance_diagrams>
        <instance_diagram_example diagram_type="EXPRESS-I">
                <template_example_proxy name="actual_part"/>
                <figure id="actual_part_inst_tmpl">
                        <title>Instantiation of actual_part template</title>
                        <img src="images/actual_part_inst_tmpl.png"
                        master="dvlp/actual_part__inst_tmpl.giz"/>
                </figure>
        </instance_diagram_example>
</instance_diagrams>
```

3.4.13.4 Instance diagrams: Instance diagram

3. The png file containing the instance diagram for the business template shall be placed in the "dexlib\data\busconcept\TLSS\templates*$template_name*\images" directory and named in accordance with the src attribute of the <*img*> element above.
4. The instance diagram in the "$template_name_inst.png" file should follow the example below:

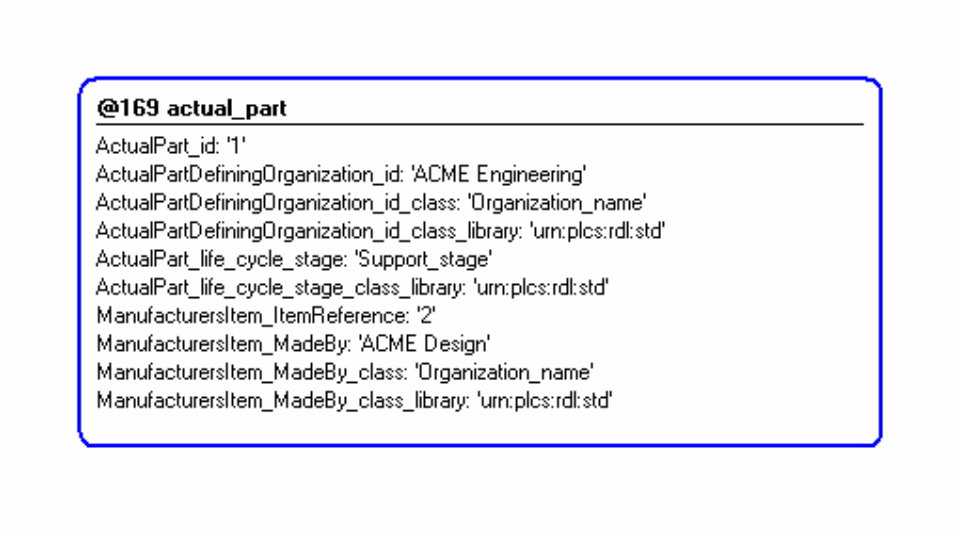

Figure 8 Instance diagram example for the actual_part Business Template

This template instance diagram is the content of the instance_diagram_example element and is named actual_part_inst_tmpl.png. It represents a populated example of the template drawn as a single box.

3.4.14 Characterizations

1. All optional attributes and relationships of a TLSS business object shall be defined as Characterizations of the corresponding business template. They should not be included on the template model diagram.
2. The entry for the characterizations element in the "template.xml" file should follow the example below:

<characterizations>
<characterization name="Creation date" optional="yes">

The date when the actual_part was created can be represented by assigning a date (using the relationship
<express_ref linkend="date_time_assignment:arm:Date_time_assignment_arm.Date_or_date_time_assignment"/>)
to
<express_ref linkend="product_as_individual:arm:Product_as_individual_arm.Product_as_individual"/>
using the
<template_refname="assigning_calendar_date" capability="assigning_date_time"/>
template with the
<express_ref linkend="date_time:arm:Date_time_arm.Date_time"/>
classified as a type of
<rdl_ref id="Date_actual_creation" urn="urn:plcs:rdl:std"/>.

<note>
The assignment of dates is described the capability
<capability_ref linkend="assigning_date_time"/>.
</note>
</characterization>
<characterization name="Date UID marked" optional="yes">

The date when the UID was marked on the actual_part can be represented by assigning a date (using the relationship
<express_ref linkend="date_time_assignment:arm:Date_time_assignment_arm.Date_or_date_time_assignment"/>)
to the <express_ref linkend="identification_assignment:arm:Identification_assignment_arm.Identification_assignment"/>
that references
<express_ref linkend="product_as_individual:arm:Product_as_individual_arm.Product_as_individual"/>
using
<template_ref name="assigning_calendar_date" capability="assigning_date_time"/>
template with the
<express_ref linkend="date_time:arm:Date_time_arm.Date_time"/>
classified as a type of
<rdl_ref id="Date_actual_creation" urn="urn:plcs:rdl:std"/>.

<note>
The assignment of dates is described the capability
<capability_ref linkend="assigning_date_time"/>.

```
        </note>
    </characterization>
</characterizations>
```

4 TESTING THE TEMPLATE DOCUMENTATION

4.1 Test the DEX files on the local machine

1. Once the "template.xml" file is complete and all associated diagrams and contacts have been completed, the files need to be tested, first on the local machine of the developer.
2. Each "template.xml" file shall be checked for is wellformedness.[10]
3. Each "template.xml" file shall be checked for validity against the relevant DTD.[11]
4. Each "template.xml" file shall be checked for DEXLib error messages when opened with Internet Explorer on the developer's local machine. To do this, right click on the "home.xml" file in the "dexlib\data\busconcept\TLSS\templates\actual_part" folder and select "Open with -> Internet Explorer" from the drop down lists. Figure 9 shows the screen that should appear.

Figure 9 Screen shot of the DEXLib view of a TLSS Business Template

[10] Note that the use of an xml editor during development will highlight errors of this kind.

[11] Note that the use of an xml editor during development will highlight errors of this kind.

5. The developer should then select every element of the Template on the top menu bar and visually inspect the content that is displayed. There should be no errors – in red text – anywhere in the displayed text. If there are, they shall be investigated and rectified.[12]

4.2 Upload the DEX files to Sourceforge

1. Once the "template.xml" and associated files have been tested on the local machine, they all need to be uploaded to Sourceforge using wincvs, See DEXlib help file http://www.plcs-resources.org/plcs/dexlib/help/dex/dvlp_intro.htm
2. Note that you must perform the process in the following order:
   a. Run wincvs.
   b. Add the folder containing all the Business Template files.
   c. Add each of its subfolders (\dvlp, \images, \sys).
   d. Add each file within each folder and subfolder and then Commit it.
3. When complete, all folders should have a tick in their icon to indicate that they have been added to cvs, and no files should have question marks left in their icon:

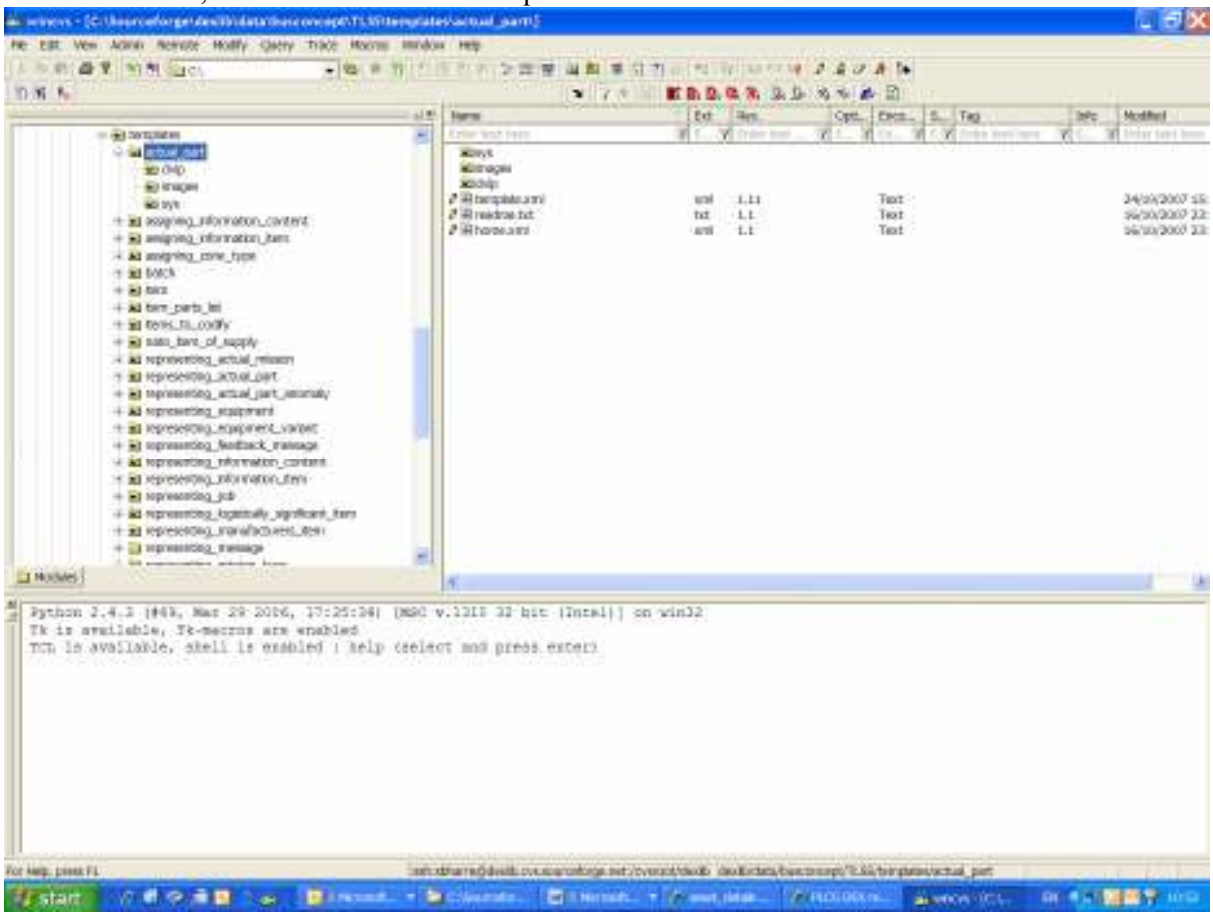

Figure 10 Screen shot of wincvs showing the file structure and icons on completion of uploading of Business Template files to Sourceforge

[12] Note that a Checklist for reviewing PLCS Templates has been developed and this may in the future be modified for use with TLSS Business Templates.

The current document is "dexlib\docs\Checklists\ Template_Review_Checklist.doc"

4.3 Test the DEX files on the "plcs_resources" web site

1. A process has been put in place that builds an HTML version of DEXLib overnight and makes this available on a public web site for read only access. This web site is http://www.plcs-resources.org/.
2. Hence the files uploaded by developers are processed during the early hours of the morning following their upload to Sourceforge. This processing can uncover errors in the source files that are not obvious upon visual inspection in Internet Explorer on the developer's local machine.
3. Consequently, all TLSS DEX developers shall perform a second check of their DEX by repeating the inspection described in section 4.1.
4. To obtain access to the HTML version of the DEX go to http://www.plcs-resources.org/
   a. Select the "DEXs" tab (second from left at the top of the screen).

Figure 11 Screen shot of the plcs-resources web site front page

   b. Select the "development release of DEXLib" link on the right hand side of the screen.

Figure 12 Screen shot of the plcs-resources web site DEXs page

c. Click the link that is then displayed – this will indicate the date and time of the HTML build, which should always be today's date (if it isn't, something has prevented the HTML build from completing – contact the Eurostep Limited Office to report the problem (+44 (0) 1745 582008)).

Figure 13 Screen shot of the plcs-resources web site DEX Development Release link page

d. Navigating to the Business Template that has been created ("Business DEXs" button on the top of the DEXLib toolbar, followed by the "TLSS" menu item on the left, then the "Templates" menu item and then the menu item for the named Business Template).
e. Now repeat the visual check of the Template selecting every element of the Template on the top menu bar and visually inspect the content that is displayed. There should be no errors – in red text – anywhere in the displayed text. If there are, they shall be investigated and rectified.

5 APPROVAL OF TLSS BUSINESS TEMPLATES

1. Each TLSS Template shall be reviewed by those named in section 3.4.2.
2. Issues arising from the review process shall be raised by the reviewer by editing the "dexlib\data\busconcept\TLSS\templates\actual_part\dvlp\issues.xml" file and uploading this to Sourceforge.
3. Review of TLSS Templates is a three stage process.
    a. The first review stage – the business review – is done when the *<business_perspective>* and *<business_object_definition>* elements have been completed. When complete and accepted, the *status* element should be updated as follows:

    <status state="in_work" completion_date="2008-01-28">

```
        <review
                type="business"
                reviewer="andy.burden"
                status="end_business_review"
                completion_date="2007-11-31"
    />
    </status>
```

b. The second review stage – the model review – is done when all parameters, paths, instance diagrams, examples and associated reference data have been fully defined. When complete and accepted, the *status* element should be updated as follows:

```
<status state="in_work" completion_date="2008-01-28">
        <review type="model"
                type="business"
                reviewer="andy.burden"
                status="end_business_review"
                completion_date="2007-11-31"
        />
        <review type="model"
                reviewer="timturner"
                status="end_model_review"
                completion_date="2007-12-15"
        />
</status>
```

c. The third stage – completion – is done when the DEX has been completed and all issues have been resolved. When complete and accepted, the *status (state)* element should be updated to "complete" as follows:

```
<status state="complete" completion_date=2008-01-28">
        <review type="business"
                reviewer="andy.burden"
                status="end_business_review"
                completion_date="2007-11-31"
        />
        <review type="model"
                reviewer="timturner"
                status="end_model_review"
                completion_date="2007-12-15"
        />
</status>
```

4. When a DEX reaches "completed" status it shall be released using the DEXLib Tool.

6 MAINTENANCE OF TLSS BUSINESS TEMPLATES

1. Once a Template has been released, all elements of it come under configuration control. This means that changes need to be proposed and approved before being implemented.

7 REFERENCES

[1] TLSS Data Exchange Specification Development Methodology.
[2] Guidance on writing TLSS business objects.
[3] TLSS Reference Data Development Methodology.
[4] Guidance on creating TLSS Reference Data.
[5] AP239 Mapping Guidelines.
[6] ISO/TC 184/SC4 Organization Handbook; Clause 3.3 Standard Enhancement and Discrepancy System (SEDS).