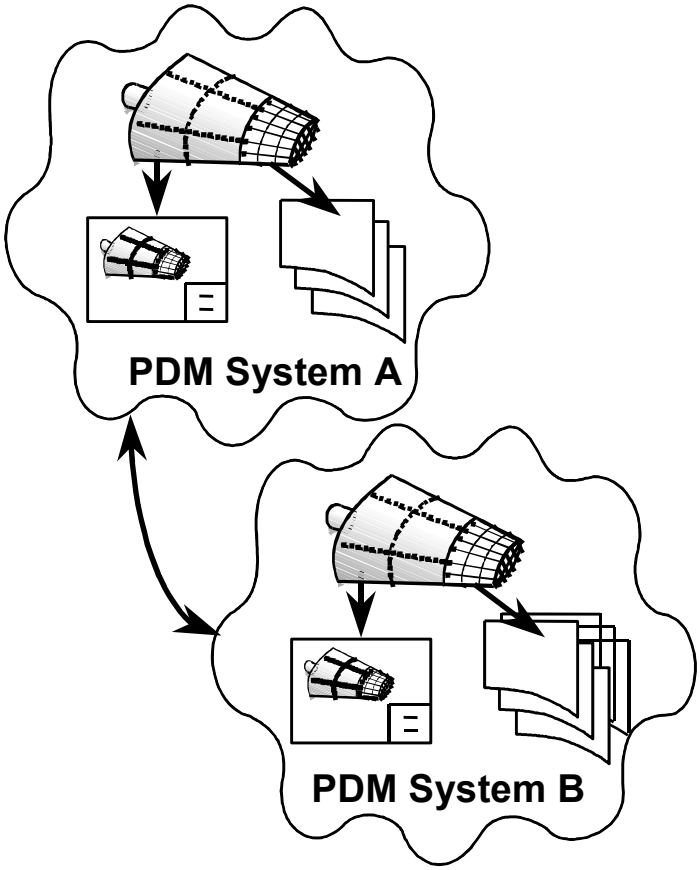
DRAFT
Recommended Practice Guide
for
Data Packaging Core Information and Exchange
(Application Protocol 232)
Version 1.0

March, 2002



Contacts:

Glen Ziolko
Northrop Grumman Information Systems
P.O. Box 224887
Dallas, TX 75222-4887
USA
972-946-4664
ziolkgl@mail.northgrum.com

Floyd Ganus
Northrop Grumman Information Systems
P.O. Box 224887
Dallas, TX 75222-4887
USA
972-946-3191
ganusfl@mail.northgrum.com

Advanced Technology Institute
The Distributed Technology Management Company

FOREWORD

This version of the Recommended Practices Guide for Technical Data Packaging provides guidance for implementing the parts list and indentured data list conformance classes in ISO 10303-232. This guide will be expanded to include the other conformance classes. Great effort has been made to harmonize with other ISO 10303 application protocols. Harmonization is being captured in the PDM schema Usage Guide that PDES Inc. and ProSTEP have been creating. This recommended practice guide adds to the results of that effort to include additional functionality. The recommended practice for the exchange management conformance class, called data definition exchange, is also included in this version of the document. There are references to modules in this document that refer to implementable chunks of functionality that are being documented by PDES Inc. The modules referenced are based on the PDM schema and AP 232.

TABLE OF CONTENTS

LIST of FIGURES

ACRONYMS

assy	assembly
CC	conformance class
comp	component
def	definition
doc	document
DPE	document_product_equivalence
dwg	drawing
ed	related
IDL	Indentured Data List
ing	relating
P	Product
PD	Product_definition
PDF	Product_definition_formation
PDFR	Product_definition_formation_relationship
PDM	Product Data Management
PDR	Product_definition_relationship
PDWAD	Product_definition_with_associated_documents
PL	Parts List
prop	property
PRPC	product_related_product_category
ref	reference
rep	representation
RPG	Recommended Practice Guide
spec.	specification
sub-assy	sub-assembly

Definitions

Technical Data Package (TDP) - a group of technical data that is collected and managed for some business purpose. Part of the business purpose can be for exchange among enterprises.

Technical Data Package Element - an element of a Technical Data Package. These elements are typically documents that are managed at some level of configuration.

Item – The primary things that is produced, controlled, sold or consumed by an enterprise. Items are things like assemblies and components.

1 Introduction

This Recommended Practice Guide (RPG) provides basic direction for the implementation of ISO 10303-232 IS, which is a STEP application protocol that provides capability to exchange product data among product data management (PDM) systems. The information that is exchanged and shared is formulated into groups based on business defined needs. These groups of information are identified, managed and related to one another. These groups of information can be correlated to the concept of a document. **AP 232 uses the term element to identify these groups of business defined data.** The types of documents for which AP 232 defines data structures are exchange management, parts list, indentured data list, data list, index list, other list, and product data sets (geometry). Data Structures for identification of drawings and drawing sheets are also defined. Each one of these types of documents can be exchanged individually, in combination in a file or shared across PDM systems. AP 232 brings to STEP the new capability of specifications to capture both the contents and configuration parameters of a document

Each type of document has a section in this RPG that describes the type of information it handles. EXPRESS constructs that are required to capture this information are defined with string constraints identified. The mapping table in ISO 10303-232 IS standard is required to view the entire path that needs to be instantiated. The diagrams provide guidance in interpreting these mapping tables.

There are three basic types of technical data packaging that capture different levels of exchange management: management of simple file exchange, management of a set of documents and their representative files, and management of product structure that contains relationships among parts and documents with their representative files. This third type can be thought of as an indentured list of things with parent-child relationships among them. The third type, product structure, is also subdivided into three indentured list methods. These are indentured list by document, indentured list by part, and indentured list by part with document references to parts. These three indentured list method are described in their own section.

2 Scope

This version of the Recommended Practice Guide will only address the exchange management, parts list, and indentured data list aspects of AP 232. The exchange management portion provides the information to manage the exchange process of document, files and product structure among different enterprise PDM systems. This version of the RPG will address each of the three exchange levels methods of identifying and managing elements of an exchange.

3 General Recommended Practice Guide Rules

For entity attributes in the schema that are OPTIONAL with no information is provided by the preprocessor, the value should be $. For entity attributes in the schema that are not OPTIONAL with no information is provided by the preprocessor, the value should be '/NULL' or ''. The distinction between '/NULL and '' is based upon the reason no information is provided. The recommendation is that '/NULL' be used by a sending system that does not manage this information, and '' be used by a sending system that manages the information but does not include it in the exchange information.

4 Document Identification, Change and Configuration Properties

All of the conformance classes share in common a number of ARM constructs and the resulting mapping in the "Mapping table for common". This section provides basic directions for implementing that portion of the common mapping table related to document identification, change identification and configuration properites. The terms element and document will be used interchangeablely in this discussion. The definition of element in AP232 is a group of product information of interest. This is not much different from a document which is defined as 'a writing conveying information'. However a document usually has the conotation of being presented physically on a piece of paper and this standard is concerned with computer data. This distinction is at the heart of this standard which is to model the information which is transmitted on paper documents (or the image of these) such as parts lists, data lists, etc and show how to transistion these to computer interpretable data.

4.1 Document Header

A document header is the configuration and data management information necessary to identify and manage a document and was normally shown in a document header. The document header contains information regarding document identification and configuration properties. The document header module is generic for any document. The ARM model for the header and header_configuration_with_element_identification are shown in Figure 4.1 ARM Model for Header.

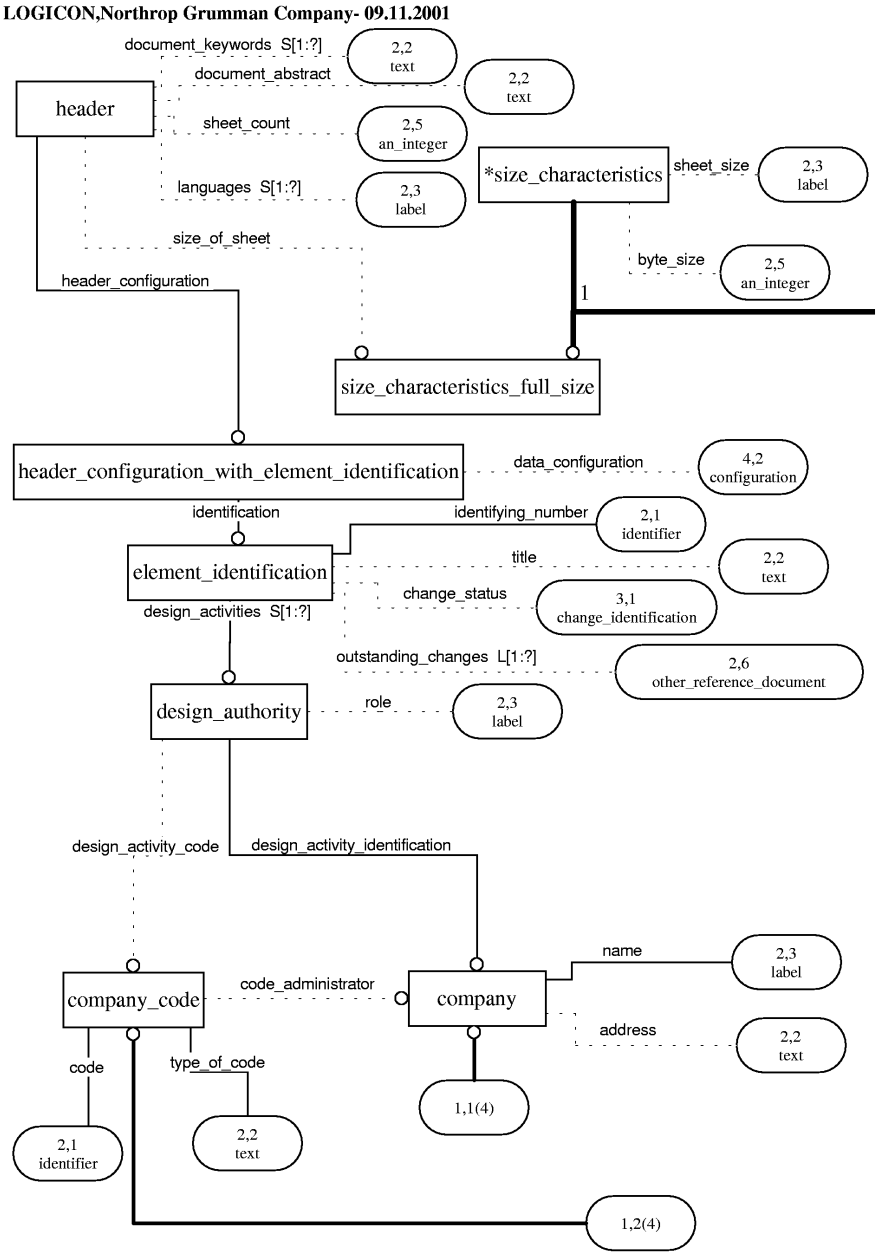


Figure D.1 - AP232_ARM EXPRESS-G diagram 1 of 4

Figure 4.1 ARM Model for Header

The parameters that make up the document header are the following

— Configuration properties;

— Document identification (identifying number, title, classification);

— Document size (sheet count, size of sheet);

— Design authority;

— Outstanding changes;

— Change identification (See separate section 4.2).

Figure 4. 2 shows how these parameters are instantiated together.

4.1.1 Configuration Properties

Configuration properties are properties that support the management of the configuration or the document relative to its data usage rights, end item system designation, security identifications, release authorizations, contracts preparation, and distribution authorizations. Configuration properties are in a separate module, section 5.3.

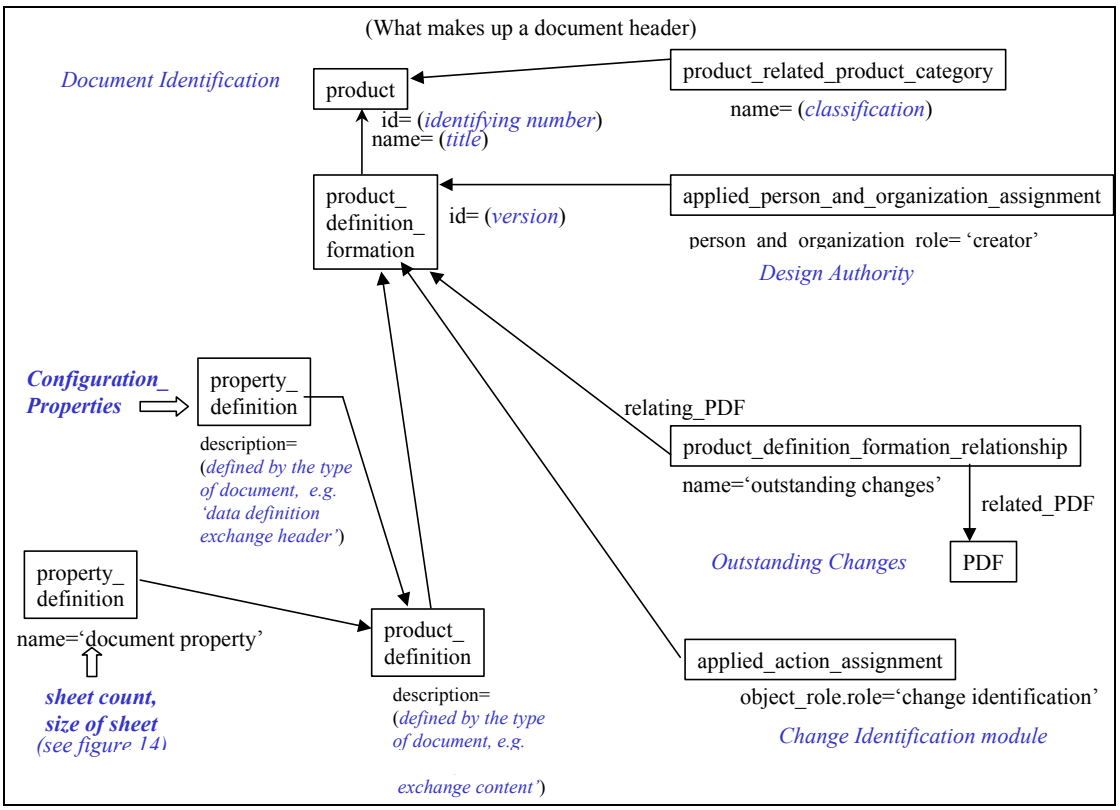


Figure 4. 2 - Document HeaderDocument Identification (identifying number, title, classification)

Document identification is made up of a collection of parameters. The identifying number is a combination of an identifying string of characters (**product.id**) and change identification (see 4.2). The title of the document is captured in the **product.name** attribute. A document is classified using a **product_category**. Initially to distinguish a document from a part a product_-related_product_category is instantiated with its name attribute constrained to 'document' . To make further clarifications of what type the document is can be done with a hierarchy of **product_related_product_category'**s using **product_category_relationship**'s. Any example of this is shown in Figure 6.25. See Section 7 of the PDM-User's Guide.

4.1.2 Document Size – (sheet count, size of sheet)

Document size as it applies to the document header is made up of 2 parameters, sheet count and size of sheet. Sheet count is the number of pages in a document. Figure 6.17 shows how to instance a sheet count. On Figure 6.17, sheet count is denoted as page count. Size of sheet is the physical or viewable dimension of a sheet. Figure 6.19 shows how to instance a size of a sheet. On Figure 6.19, size of sheet is denoted as size format.

4.1.3 Design Authority

Design authority is the person and organization, or organization, that is the owner of the document version. The person and organization module is used here with **applied_person_-and_organization_assignment** pointing to PDF of the document. The role of the **person_and_-organization_assignment** is contained in **person_organization_role.name**='creator'. The author of a document is the same as in design authority for a part. Design Authority is a required to uniquely identify a document. AP232 also allow the creator role name to be 'owner id', but creator is recommended for compatibility to PDM schema user's guide.

4.1.4 Outstanding Changes

Outstanding changes are documents that contain change information that is not incorporated into the related document. These outstanding changes can be supplements and are used in conjunction with the related document.

EXAMPLE An example would be a drawing that has a couple of drawing change notices that are used together to define the latest version of a part. The document representing the outstanding change would be instantiated using the document identification module (product, PDF, PRPC). The relationship between the outstanding change document and the document the outstanding change applies to is captured using a **product_definition_formation_relationship** (PDFR). The **PDFR.relating_product_-definition_formation** would point to the document the outstanding change applies to and the **PDFR**.**related_product_definition_formation** would point to the outstanding change document.

At first this may seem to conflict with the PDM schema user's guide. However, outstanding changes is the document definition of what changed while the PDM schema user's guide is identifying the change process.

4.2 Change Identification

Change identification provides the ability to identify a change in information to a part, document, or file. Change information provides the ability to identify:

- different levels of change (e.g., revision, change, issue, sequence),
- description of the changes;
- the dates these different levels of change became effective;
- a change code based on the collection of changes that have been placed on or incorporated into the document;
- the status of the change process;
- revision authorization documentation.

Figure 4.3 and Figure 4.4 shows how these change information parameters are instantiated.

The identification of the version only of a part or document is captured through a product_-definition_formation.id as defined in the product identification module. When more than version is to be identified, i.e. change level, issue or sequence, then the change identification will be an executed action.

Change identification information is collected through an executed_action. Executed_action was selected because the change being identified is the result of an action that has been executed to some level at a particular point in time. As a product goes through its life cycle many changes or modifications may be placed on or incorporated into it. Each change has a specific level of impact to an enterprise and the product it affects. The magnitude of impact guides the enterprise to place a particular change level to a product. The needed changes to a product can also be identified and applied to the product while the incorporation of the change into the documentation will be done at some later point in time. This condition is some times referred to as Outstanding Changes. Identifying a document that has outstanding changes is also a requirement.

The use of exectued_action to define change identification is a capability defined beyond what is given in the PDM Schema User's Guide. The PDM Schema User's Guide is limited to a single level of change, i.e. revision.

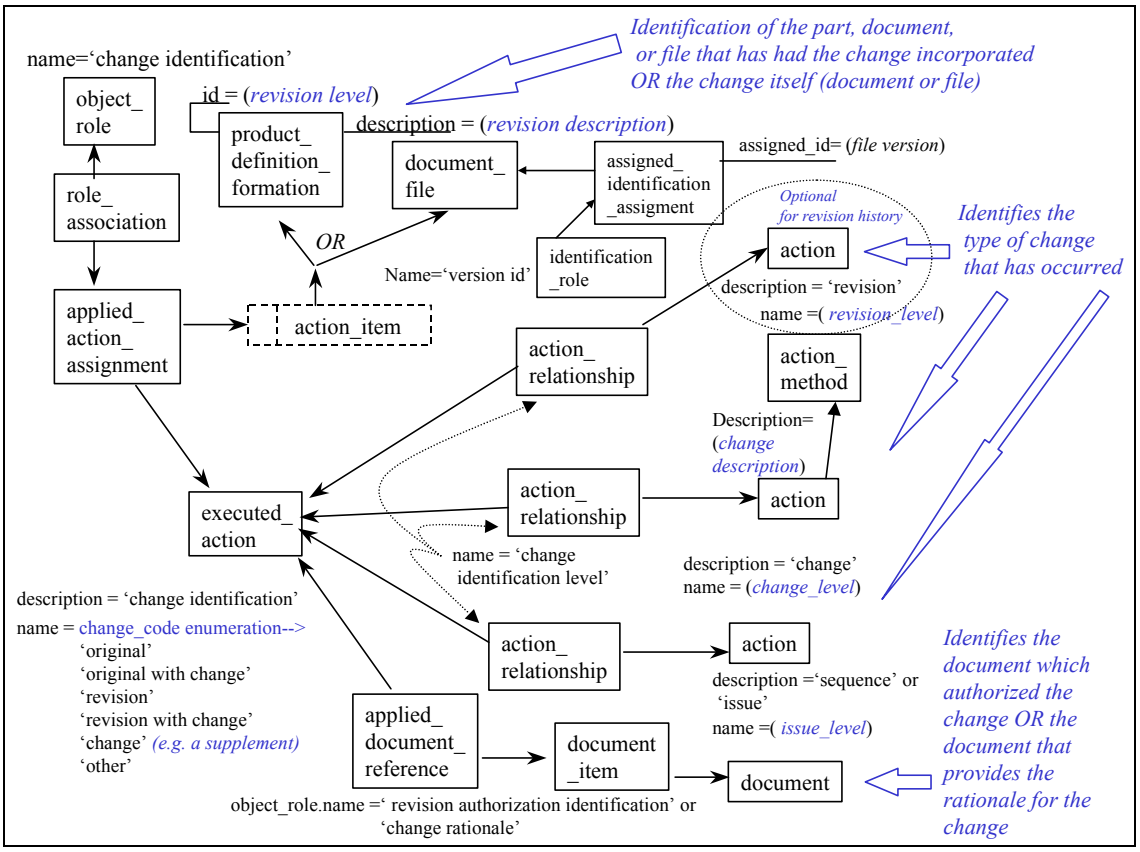


Figure 4.3 - Change Identification (1 of 2)

EXAMPLE An example of an enterprises change policy could be a change to a document may affect the part that is documented in the document. If the form, fit, or function of the part changes, then the part number changes. If a clarification to a part definition is documented on a change to the document, then the part number does not change. If a process that defines the part changes and it is documented as a change to the document, the part number does not change, but the document changes. The fundamental concept is that any change to a document will roll forward the document revision level. If the form, fit, or function of a part changes, then the part number will change. There are a lot of things that are related to a part that can change, without changing the part number, but will change the document for that part.

4.2.1 Different Levels of Change

A project or an enterprise establishes the different levels of change. The four levels of change defined here for use are the following:

— A sequence level change to a document or part is performed just by the fact that it has been checkout in a PDM system (Given changed rights to the person who checked the document or part out). A sequence level can result in the part or document being changed at any of the three other levels.

— An issue_level is a change to a document or part that has no affect on the part that is defined on the document (e.g., fix spelling error, reformat information, change fonts, move things around).

— A change_level is a change to a document or part that was based on a change to a part that did not change form, fit, of function (e.g., paint color changed, serial number stamped on part, pilot hole added).

— A revision level is a change to a document or part that was based on a change to a part that did change form, fit, or function (e.g., dimensional tolerance change, new length, different material).

There may be some exceptions to the above levels of change, which should be negotiated between business partners.

A level of change different from these should be established between partners prior to the exchange.

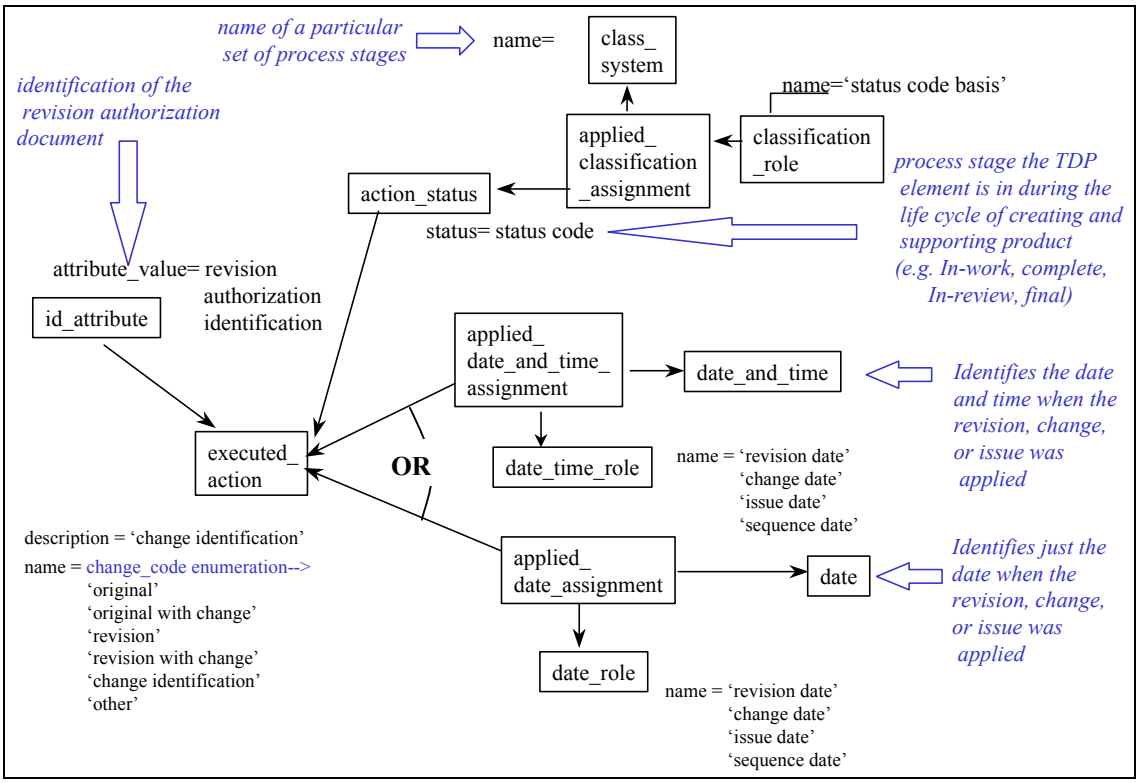


Figure 4.4 - Change Identification (2 of 2)

4.2.2 Dates of Change

For each of the change levels a date may be applied, identifying when the change was applied.

4.2.3 Change Codes

The change codes identified what set of changes apply to a product. The change codes are the following:

— Original;

— original with change;

— revision;

— revision with change;

— change;

— other.

4.2.3.1 change
The change_identification is a File, Item, or Tdp_element that modifies an original, revision, or supplement.

NOTE Revision level is typically not defined for an original or is defined as zero. The change level is the highest level change included.

4.2.3.2 description of change
The description of the changes provide details on what actual changes occurred.

4.2.3.3 original
Tthe File, Item, or Tdp_element that is the original release.

4.2.3.4 original with change
The File, Item, or Tdp_element that is the original with changes incorporated.

NOTE Revision level is typically not defined for an original or is defined as zero. The change level is highest change included.

4.2.3.5 other
A modification where none of the other Change_codes apply.

4.2.3.6 revision
The File, Item, or Tdp_ element that is a complete revision to a File, Item, or Tdp_element.

NOTE Revision_level indicates current revision level. Change_level is typically zero for a complete revision.

4.2.3.7 revision with change
The File, Item, or Tdp_element that is a complete revision of a document or product data set with changes incorporated.

NOTE Revision_level is current revision level. Change_level is highest change included.

4.2.4 Status of Change Process
The action_status specifies the life-cycle stage or the working-process stage of the Tdp_element being identified. The action_status need not be specified for a particular change_identification.

NOTE Action_status is the identification of the process stage the Tdp_element is in during the life cycle of creating and supporting product information.

EXAMPLE In-work; Complete, In-review, Final, are examples of action_status.

4.2.5 Revision Authorization Identification

The revision authorization identification specifies an identifier or an other_reference_document that is the identification of the revision authorization document. The revision_authorization_ identification need not be specified for a particular change_identification.

Note 1 In industrial practices, a revision authorization document is used in lieu of a revision description or a revision record.

Note 2 The revision authorization document may describe the revision history of the file, item, or Tdp_element since the original release of the File, Item, or Tdp_element.

Note 3 The revision authorization document may describe a single revision of the File, Item, or Tdp_element.

Note 4 The Other_reference_document specifies that the Revision_authorization_select is a Reference_ document that defines the rationale for the Change_identification.

Note 5 The identifier specifies that the Revision_authorization_select is other than an Other_ referenced_document and identifies the authorizing article.

4.3 Configuration Properties

Configuration properties are properties that support the configuration management of parts and documents. Configuration properties consist of data usage rights, end item system designation, security identifications, release authorizations, preparing contracts, and distribution authorizations (see Figure 4.5).

The configuration properties of data usage rights, end item system designation and security identifications are described in this section. The configuration properties of release authorizations are captured in section 5.3.2, preparing contracts in section 5.3.3, and distribution authorizations in section 5.3.1.

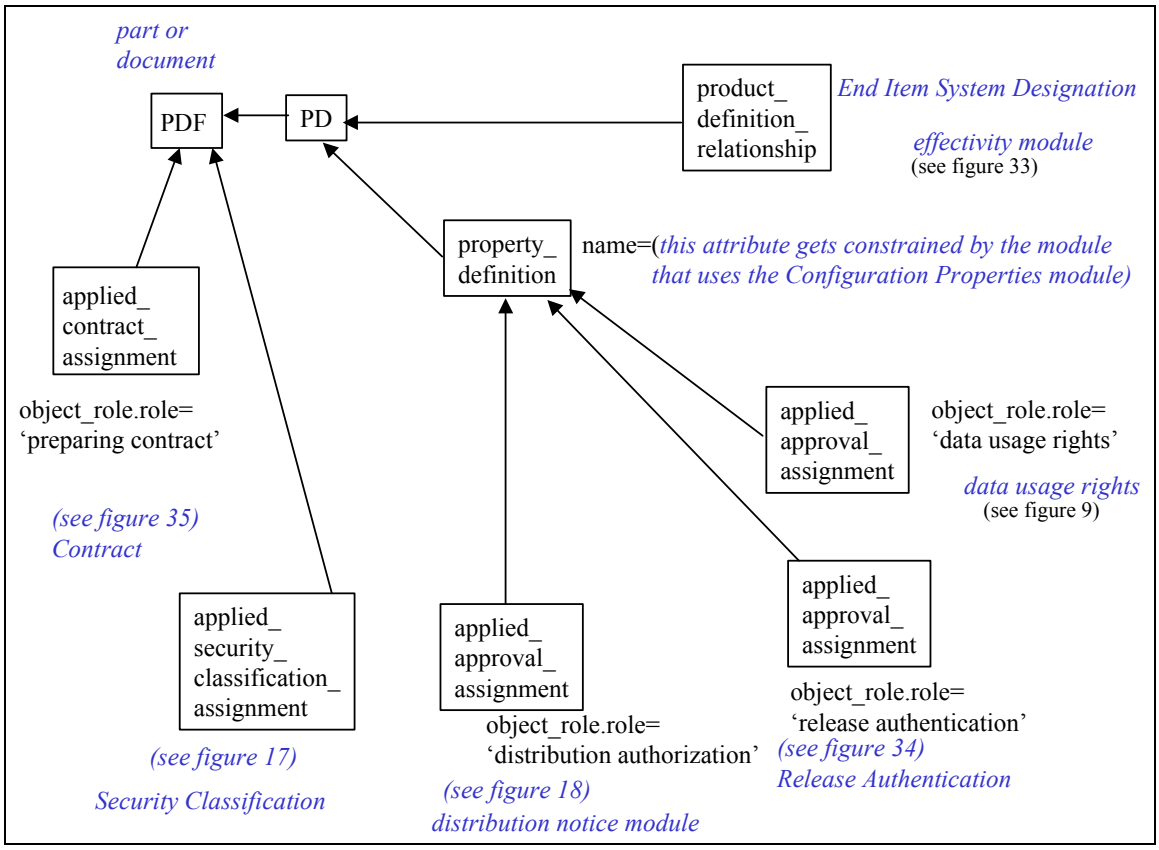


Figure 4.5 - Configuration Parameters

4.3.1 Data Usage Rights

Data usage rights are the legal rights the user of the data has with this data. The data usage rights need not be specified for a particular Configuration.

EXAMPLE Limited rights indicates the user has to obtain legal rights from the originating data source to use the data. Unlimited rights indicate the user does not have to obtain legal rights to use the data.

In industrial practices, data rights (and data rights codes) can be specified within the context of the contract for which the data was prepared.

An instance of the entity **approval** is used to capture data rights information, see Figure 6.11. The approval.level attribute captures the specific text that describes the data usage rights of the data user.

EXAMPLE An example could be stipulating the data usage rights to be 'Limited'. The corresponding product_definition (for the Exchange Management document, Figure 6.1) is associated to this approval through a **property_definition** and an **applied_approval_assignment**. The corresponding attributes are constrained to uniquely identify the path.

property_definition.name= (the value here is defined in the module that uses the Configuration Properties Module,

ExampleExchange Management Header module constrains property_definition.name = ‘data definition exchange header’ which uses Document Header that uses Configuration Properties.)

object_role.name=’data usage rights’ (associated with applied_approval_assignment)
The values for approval_status.name will follow the PDM Schema RPG for approval module.

4.3.2 End Item System Designation

The end item system designation provides the link that identifies where documents and parts are effective. This property utilizes the effectivity module. Figure 4.6 shows the constructs used to satisfy End Item System Designation Property.

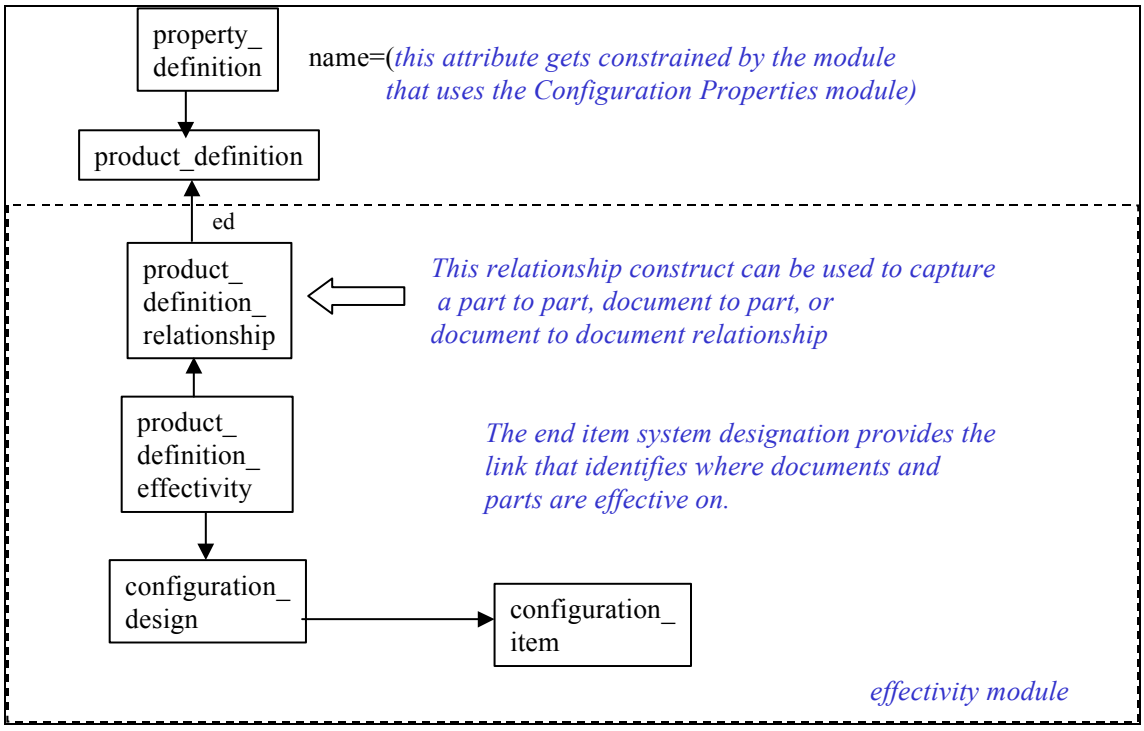
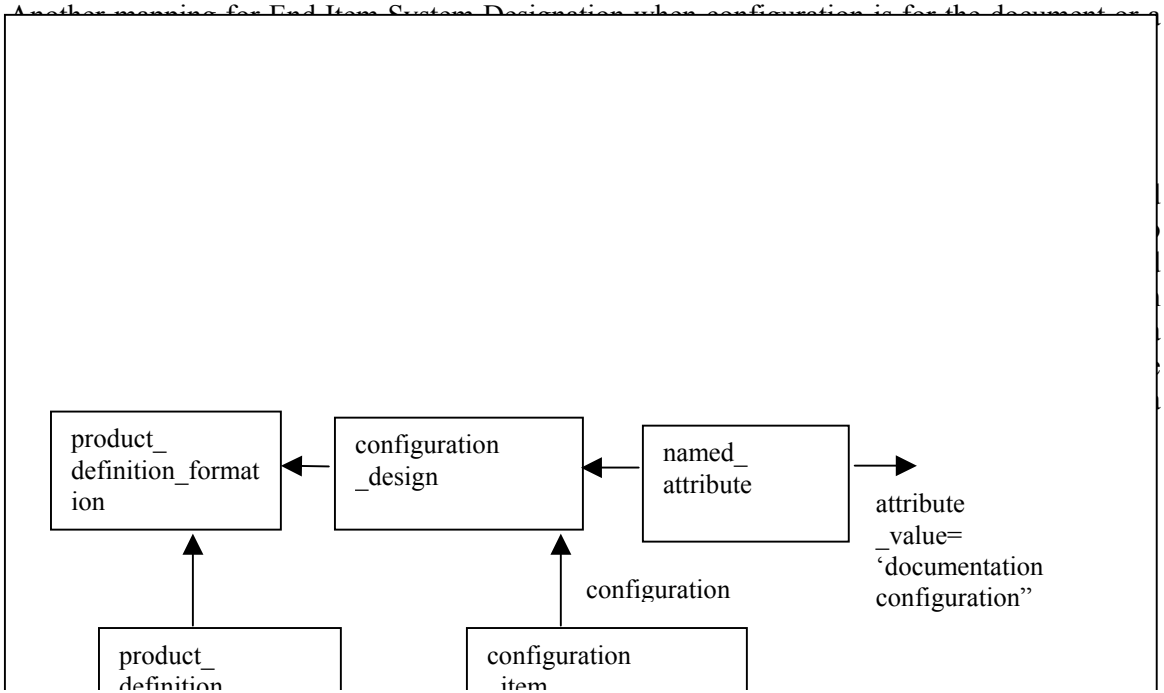


Figure 4.6 - End Item System Designation

Another mapping for End Item System Designation when configuration is for the document or a



EXAMPLE the Exchange Management Header module would add **product_definition_formation** to the **security_classification_item** select type. The Exchange Management module uses the Document Management module, which uses the Configuration Properties module.)

4.3.4 Distribution Notice Module

A distribution notice is the allowable industrial or organizational circulation of information for the organization receiving the part information, document, or file. The data that makes up a Distribution notice includes distribution authority, distribution code, and distribution statement. Each of these distribution notice elements can exist independently from one another.

Figure 6.20 shows how the distribution notice elements are captured and related.

The distribution statement specifies the distribution notice text that describes the allowed circulation. The distribution statement is captured in an **approval.level** attribute. This approval construct is related to the product data this distribution statement applies to through an **applied_-approval_assignment** with an associated **object_role.role**='distribution notice'.

EXAMPLE An example of distribution statement is "Distribution is Unlimited."

The distribution authority specifies the distribution notice controlling authority for circulation. The distribution authority is captured as an organization, or a person plus organization. This organization is associated to the approval construct that contains the distribution statement. An **approval_person_organization** construct with an associated **approval_role.role**='distribution authority' is used to associate the organization to the approval.

The distribution code specifies a code that identifies a particular condition for circulation. Many companies and industries agree on different codes to depict different conditions for circulation.

EXAMPLE An example of a distribution code could be "A" which indicates 'Distribution is Unlimited ." The distribution code is captured in the attribute **class.name**. The **applied_classification_-assignment** construct allows these codes to be applied to multiple distribution notices.

4.3.5 Release Authentication

The release authentication is the authentication of the release of the documents and parts from the originating system. The data associated with the Release_authentication are the following:

— authentication;

— person_responsible;

— release_authority;

— release_authority_code;

— release_date.

Figure 4.8 shows how these requirements are instantiated.

4.3.6 Authentication

The authentication specifies the originating system identification of the authentication.

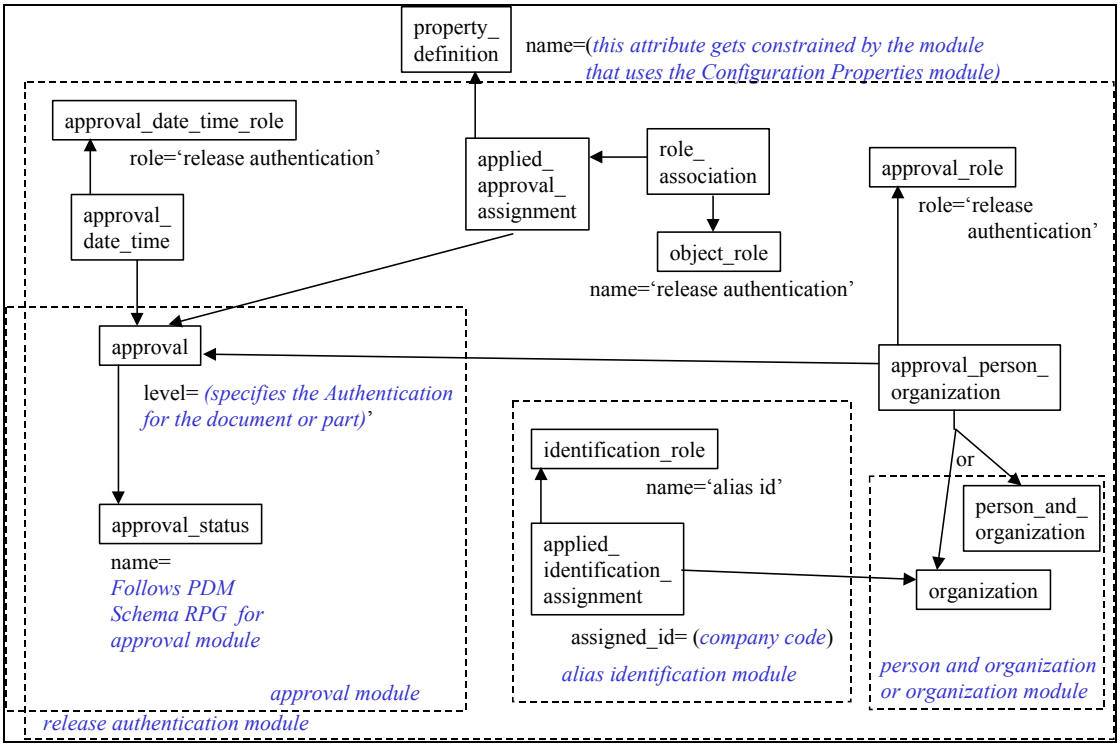


Figure 4.8 - Release Authentication

NOTE The authentication constitutes an electronic signature that assures the integrity of the Tdp_element's contents. This may be conducted by reviewing the Tdp_element's contents or by validating the automated data processing system procedures that constructed or manages the Tdp_element.

Some companies have a mixture of alphanumeric characters that they utilize to authenticate a document. Sometimes this is unique per document.

The Authentication requirement is mapped to the **approval.level** attribute. This is an additional use of the approval.level attribute that is not covered in the PDM Schema User's Guide.

4.3.7 Person responsible

The person responsible specifies the responsible person that has authenticated the Tdp_element contents for the purposes of release. The person responsible need not be specified for a particular release authentication.

NOTE In cases where a symbol is used instead of a signature, the authentication should define the symbol name. The graphical presentation form of the symbol is addressed under presentation entities.

4.3.8 Release authority

The release authority specifies the name of the Company responsible for the release of the Tdp_element.

4.3.9 Release authority code

The release authority code specifies the company code responsible for release of the Tdp_element. The release authority code need not be specified for a particular Release_-authentication.

4.3.10 Release date

The release date specifies the date and time at which the design release activity released the Tdp_element.

4.3.11 Contract Technical Data References

A contract is a legal document that describes an agreed-upon set of terms and conditions among two or more parties. In a contract there may be references to other documents and to specific data requirements. These references are handled in the contract technical data reference module. The contract technical data reference module will utilize the basic contract module.

The first requirement is to identify a contract as a document. In STEP there is a contract entity that can be assigned to product data. The contract entity is a simple identification that denotes name, kind, and purpose. To identify a contract as a document, a relationship between the contract entity and a **PDF** using the document identification module to identify the contract.document. This relationship is established using an **applied_contract_assignment** with an associate **object_role.name**=’contract document.' This is shown in Figure 4..

The second requirement is the ability to reference documents from a contract that describes data items which are to be delivered under the contract. These documents are sometimes referred to as Data Item Description (DID) documents. These reference documents are associated to the contract using an **applied_contract_assignment** with an associate **object_role.name**=’reference from contract’. This is shown in Figure 4.8.

The third requirement is the ability to specify the contract data requirement list clause or contract work listing number that authorizes, specifies, and regulates the formal agreement between two or more parties. The contract data requirements list is part of the contract. The contract_data_-requirements_list need not be specified for a particular contract. The identification of the contract data requirements list is captured using a **document_usage_constraint.subject_element_value**, with the **document_usage_constraint.subject_element**= ‘ contract data requirements list’, as shown in Figure 4.8.

Each contract data requirements list can be associated with specific parts or documents that are identified as an element in the list. These associations are achieved through the use of an **applied_document_usage_constraint_assignment**, with an associated **contract_usage_-constraint_role.name**=’preparing contract’ or ‘contract submission’. The **applied_document_-usage_constraint** points to either a PDF or a PD. A PD is pointed to when a specific

representation of a document (e.g., a specific digital format) is being designated to satisfy a contract requirement.

The capability to relate the Data Item Description documents that were referenced by the contract to the specific contract data requirements list is achieved by establishing an **applied_document_-reference** between the document construct representing the DID and the **document_usage_-constraint** representing the contract data requirements list. The associated **object_role** with the **applied_document_reference** should be constraint to **object_role.name**='base description'.

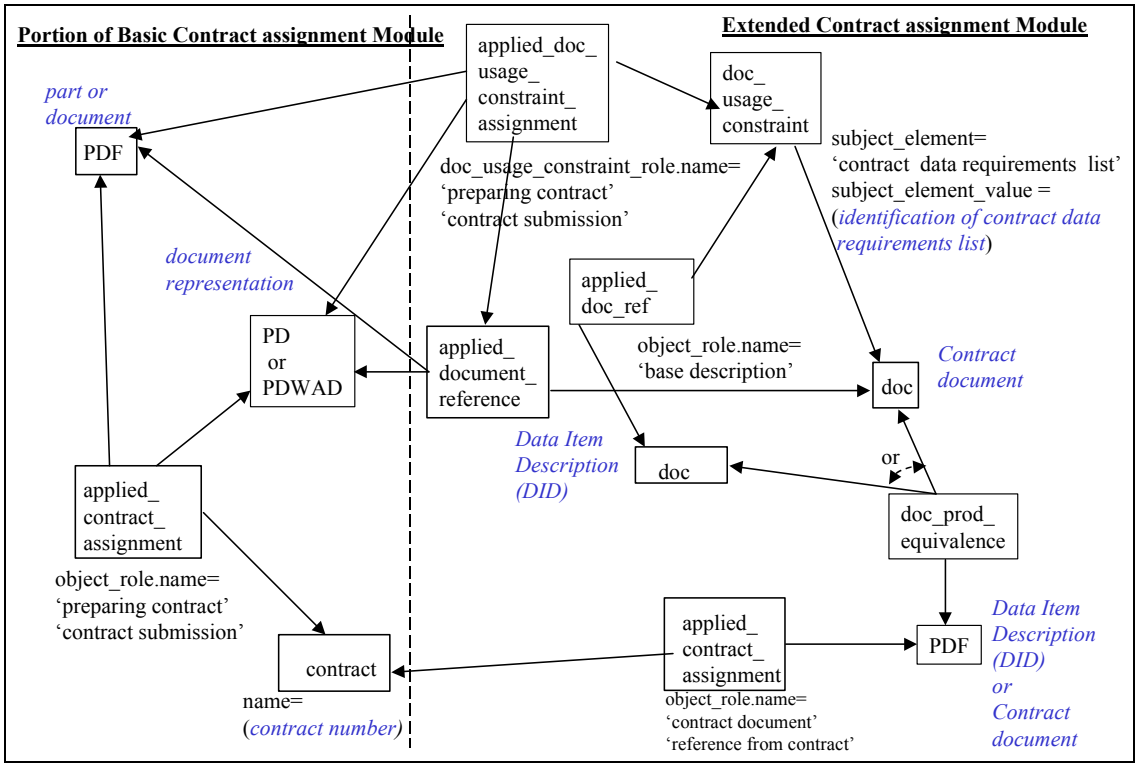


Figure 4.9 - Contract Technical Data References

5 General Information Elements and Common Rules

5.1 Notation Module

General notation that can be applied to one or many things is captured in this module. As Figure 6.11 shows, the requirement of notation is captured in the representation schema. A **representation** instance is identified with its name attribute constrained to 'notation.' This representation instance can be associated with one or many property definitions through the use of **property_definition_representation** instances.

There are four pieces of information that make up notation. The first one is the note itself. The note specifies the explanatory remarks or notations that may be useful or informative to a human. The note is captured in the **descriptive_representation_item.description** attribute.

The second piece of information is the note title. The note title specifies a textual name for the note. The note title is captured in the **descriptive_representation_item.name** attribute.

The third piece of information is a reference code. The reference code specifies an identifier that is associated with the note. The reference_code can be utilized to associate the note with a note listed on the face of the drawing, a note listed and explained in another portion of a document this note is defined in, or another listing source where detailing of notes are described. The reference_code is captured in the **class.name** attribute. **Class** is a subtype of **group** and is applied to the **descriptive_representation_item** of the note. Capturing a reference_code for a notation is optional.

The fourth piece of information is the type of notation. The type of notation is a characteristic of a note that identifies how, what, or where a notation is used. The type of notation need not be specified for a particular Notation.

Example 1 component: A component specifies that a note is referenced to a component.

Example 2 drawing: A drawing specifies that a note is referenced to a drawing.

Example 3 drawing effectivity flag note: A drawing effectivity flag note specifies that the note is flagged to reference effectivity notations.

Example 4 drawing flag note: A drawing flag note specifies the note as a flagged noted indicating that it is located in the design area of a drawing specific to one or more parts. Industrial practices utilize flag notes to flag data on the face of the drawing that are not considered in the note listing (i.e., General note) on the face of the drawing.

Example 5 drawing general note: A drawing general note specifies that the note is a general note from the face of the drawing.

Example 6 remark: A remark specifies that the note is a remark to the reference of interest.

The type of notation is captured in the **class_system.name** attribute. The **class_system** entity is a subtype of the **group** entity. This is one of the specializations of the group entity to put specific semantic against a general concept of group. **Class_system** is associated with the **descriptive_-**

representation_item by an **applied_group_assignment** with an associated **object_-role.role**='type of notation.'

5.2 Revision History

Revision history is a collection of change information that can be organized into chronological order. The amount and type of change information that can make up a set of revision history varies from organization to organization. This module identifies a variety of change information that can become part of a document or part revision history.

Change information for revision history includes the following:

— Authorizing Documents;

— Revision Approval;

— Revision Date;

— Revision Description;

— Revision Level.

Change information is accumulated in two ways. First, order for the revision history can be defined by the chaining of **PDF**s with **PDFR**s. The type of **PDFR**s are hierarchical or sequential. Each **PDF** in this collection can capture the revision level in the **PDF.id**, the revision description in the **PDF.description**. Authorizing documents, Revision approval, and Revision date can be applied to each **PDF**.

The second way is to tie the revision changed information to the actions that describe and incorporate the change. This is through the use of a collection of **executed_action**s. Each **executed_action** identifies a change. These changes can be at the same level of change or at different levels. This method allows a collection of changes to be identified together and associated to a particular version. This is done by associating each **executed_action** to a particular **PD** through the use of an **applied_action_assignment** with the associated **object_-role.role**'s change history. Each **executed_action** in this collection can capture the revision level in the **executed_action.name** or its associated **PDF.id** (with an **applied_action_-assignment.object_role.role**='change identification' same as in change identification module). The textual explanation of the revision would be captured in the required **action_-method.description**. Authorizing documents revision approval, and revision date can be applied to each **executed_action** and **PDF** (if they exist for the earlier revisions). The basic **executed_-action** method is shown Figure 5.1. A variation of this **executed_action** method is to utilize the instances of **executed_action** from the change identification module and add an additional **applied_action_assignment** to each **executed_action** that needs to be part of a revision history collection. Point these to the **PD** needing the revision history information. The dash lines in Figure 5.1 show the first **exectured_action** method.

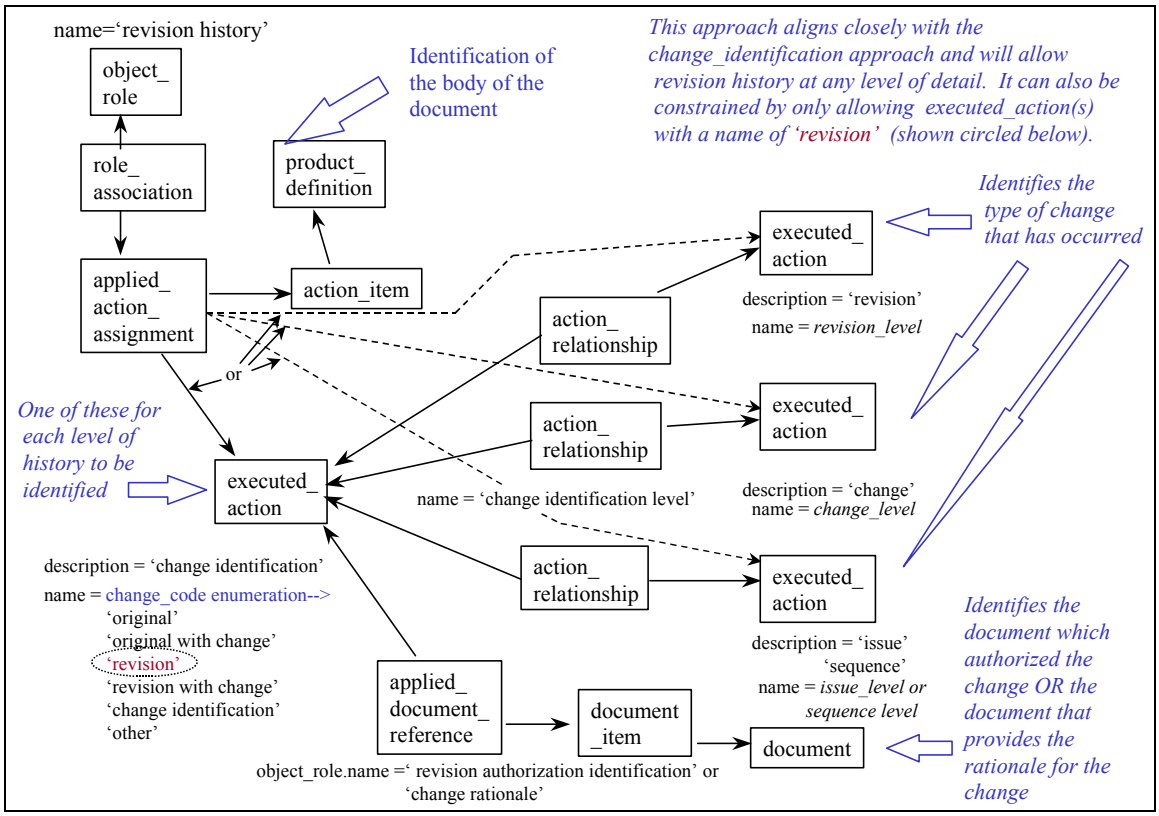


Figure 5.1 - Revision History

5.2.1 Authorizing Documents

The authorizing_documents specifies the revision authorization documents when a revision description is not provided. The revision authorization documents may describe the revision history of the Tdp_element from its original release. The authorizing_documents may not be specified for a particular revision. There may be more than one authorizing_documents for a revision.

5.2.2 Revision Approval

The revision_approval specifies the person and organization that are responsible for approving the revision. The revision_approval need not be specified for a particular revision.

5.2.3 Revision Date

The revision_date specifies the date and time that the revision_level was advanced. The revision_date need not be specified for a particular revision.

5.2.4 Revision Description

The revision_description specifies a textual explanation of the revision. The revision_description need not be specified for a particular revision.

5.2.5 Revision Level

The revision level specifies the version of the document from its original release or issue. Revisions reflect changes made to the original document after authorized release or issue that require the revision level to be advanced.

5.3 Source Identification

Source identification identifies the location or source of the product data, such as parts or documents, either through a contract or a referenced document Figure 6.12 shows this module applies to a data_definition_entry.

5.3.1 Available from - Contract Submission

A contract submission identifies a contract, the location of the contract submission, and date of submission. The contract, in Figure 6.12, is applied to the Data_definition_entry (**PD**) using an **applied_contract_assignment** with a corresponding **object_role.role** attribute constrained to 'contract submission.' The location of the contract_submision is captured with an organization entity and its organization_address entity. The organization is applied to the **applied_contract_-assignment** with the corresponding **organization_role.name**='location of contract submission.' The date of submission is captured by applying the date entity to the **applied_contract_-assignment** using an **applied_data_assignment** entity with its corresponding **data_-role.name**='date of submission'. When a time is also required to be captured with the date, apply a **date_and_time** entity to the **applied_contract_assignment** entity using an **applied_date_-and_time_assignment** with a **date_time_role.name**='date and time of submission'.

5.3.2 Available from - Document Reference

A reference document can also identify from where a document or part is available. This reference document contains the available from information. The reference document is mapped to a product and associated to the **product_definition** (e.g., Data_definition_entry) through an **applied_document_reference** with the associated **object_role.role**='source identification.' The identification of the reference document follows the document identification module RPG.

5.4 Special Conditions

A special condition is a characteristic of an item or Tdp_element that is mutually agreed on between parties for a business purpose. There are three pieces of information that make up this characteristic. The first is a code that specifies a specific identifier from a set of mutually agreeable special conditions. The string of characters that make up the code is captured in the **descriptive_representation_item.name** attribute. The second is a description of the special condition. The description is captured in the **descriptive_representation_item.description** attribute. Both the code and description information are optional but one must exist. The third is a type of coding schema. Coding schemas can be established by companies and industries that agree on different codes to depict different conditions. These coding schema agreements are sometimes contained in industry or government standards. An example of a United States standard that contains a special_condition coding schema is ANSI ASME Y14.34M. The coding schema is captured in the **representation_context.context_type** attribute. Constrained string

values are shown in Table 2. Figure 6.10 – Entry Characteristics (Include Flag, Master Rep. Flag, Special Conditions)
shows how a special condition is placed on a data_definition_entry.

5.5 Common Rules

The following rules are defined in the common table of the document and apply to all conformance classes as appropriate. These diagrams use the special notation of saying 'IF EXISTS'. When this exists the rest of the diagram should be populated as shown.

5.5.1 Approval requires approval date time

The **Approval reqires approval date time** rule specifies that each instance of approval shall be referenced by exactly one **approval_date.** See figure 5.2.

5.5.2 Approval requires approval person organization

The **approval requires approval person organization** rule specifies that each instance of **approval** shall have at least one **approval_person_organization** referencing it except for approval instances that as of collectors of other approval instances. See figure 5.2.

5.5.3 Approval are assigned

The **approval are assigned** is a recommended rule which is not in the document. This rule specifies that each instance of **approval** should be assigned by at least one instance of **approval_assignment** unless this **approval** instance that is a collector of other approval instances.

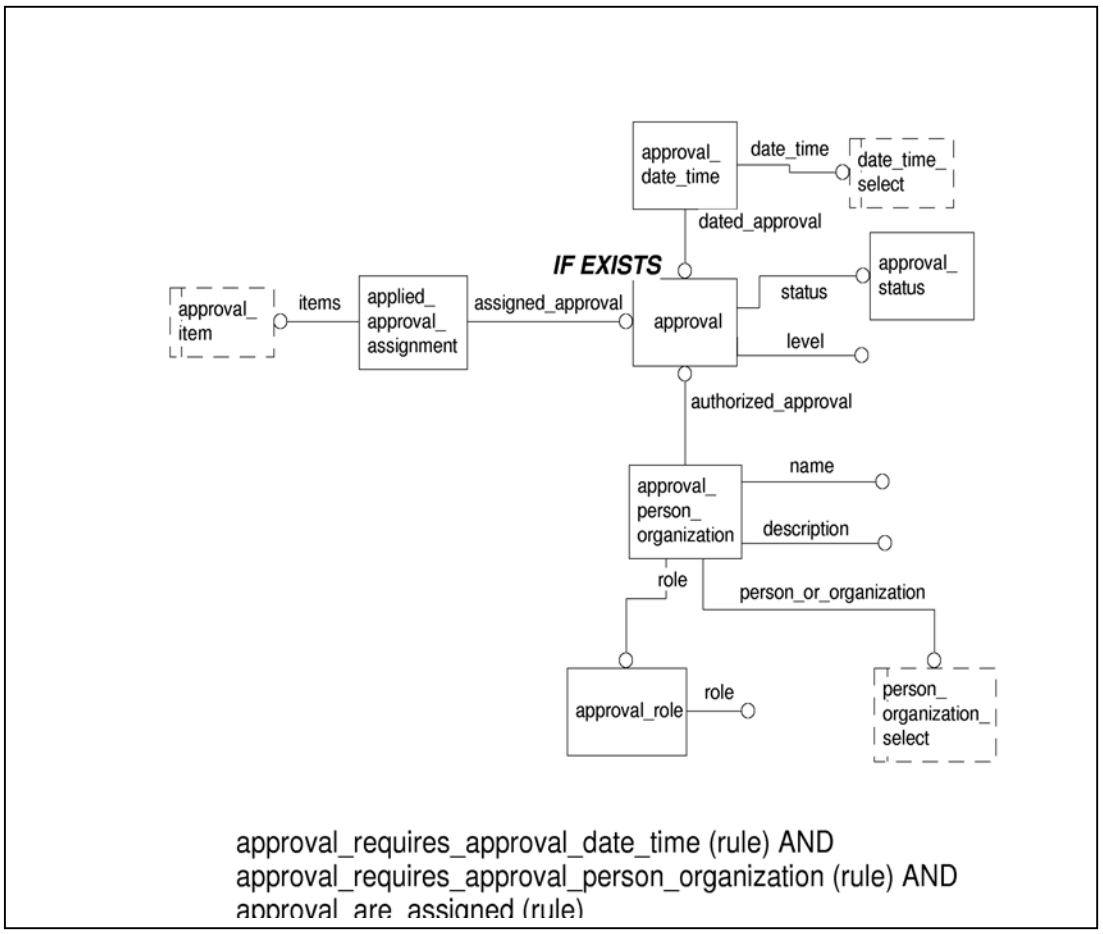


Figure 5.2 Rule Approval are assigned

5.5.4 Change identification restricts executed action

The **change identification resticts executed acition** rule specifies that an **executed_action** with **executed_action.description** equal to 'change identification' shall have one or more **applied_-action_assignment**s associated to itself.

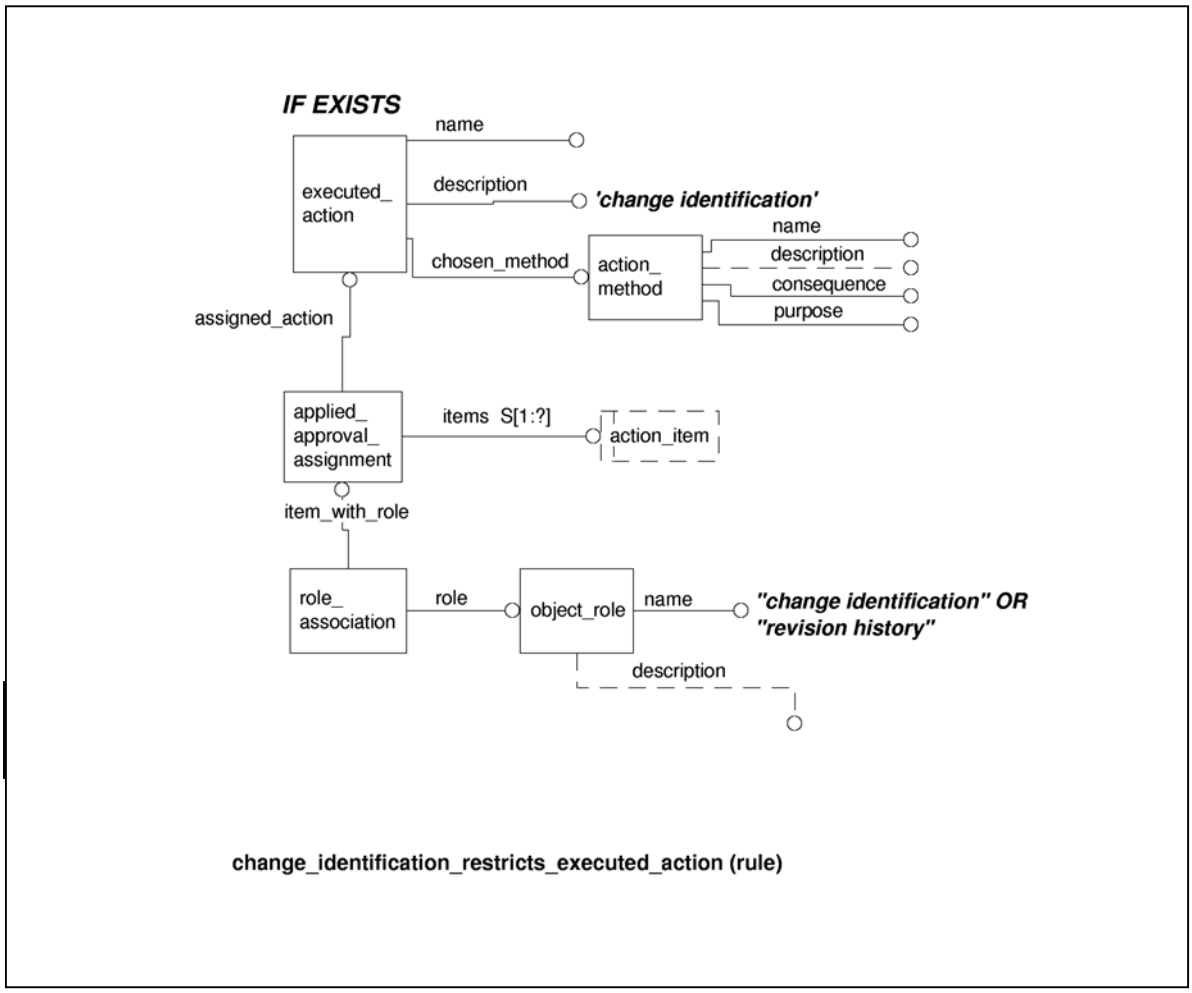


Figure 5.3 Rule Change identification restricts executed action

5.5.5 Company code string restriction

The Company code string restriction is not a rule in the document but is suggested except in those instances where the code identifying a company will be understood without this further delineation. This rule specifies how certain attribute values in identification role, organization_role and description attribute denote that a company code is being captured.

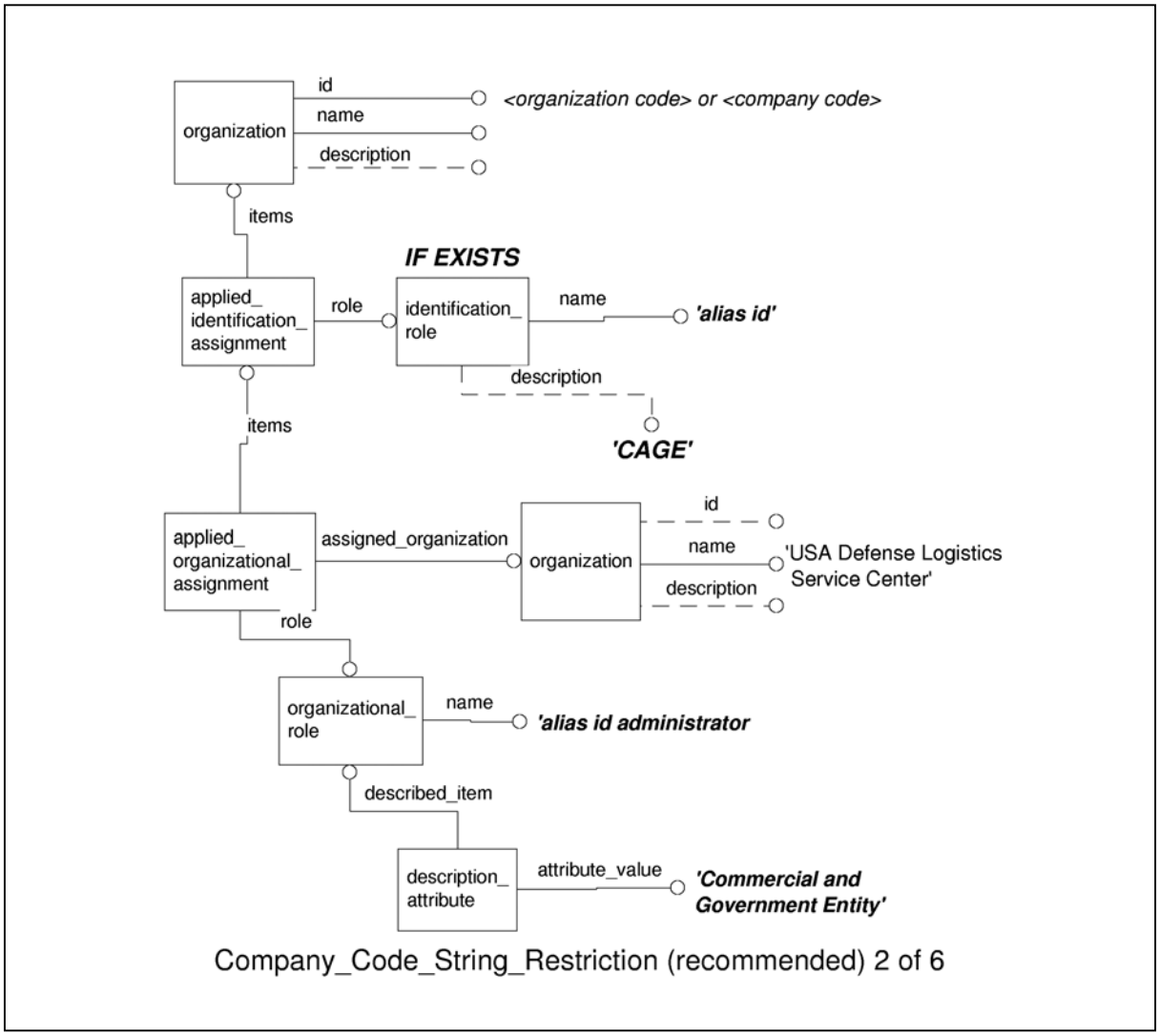


Figure 5.4 Suggested Rule Company code restriction

Figure 5.6 shows only one of five cases. The other cases are shown in the table following:

identification_role.description	description_attribute.attribute_value	organization.name
CAGE	Commercial and Government Entity	USA Defense Service Center
CEC	Contractor Estabglishment Code	
DUNS	Data Universal Numbering System	
EIN	Employer Identification Number	USA Internal Revenue Service
PASS	Procurement Automated Source Sytem	

5.5.6 Header configuration restricts property definition

The **header configuration restricts property definition** rule specfies that a **property_definition** with name equal to 'XXXheader' (with the word header being the last set of characters in the string) shall have one or more **applied_approval_assignment** or **property_definition_representation.**

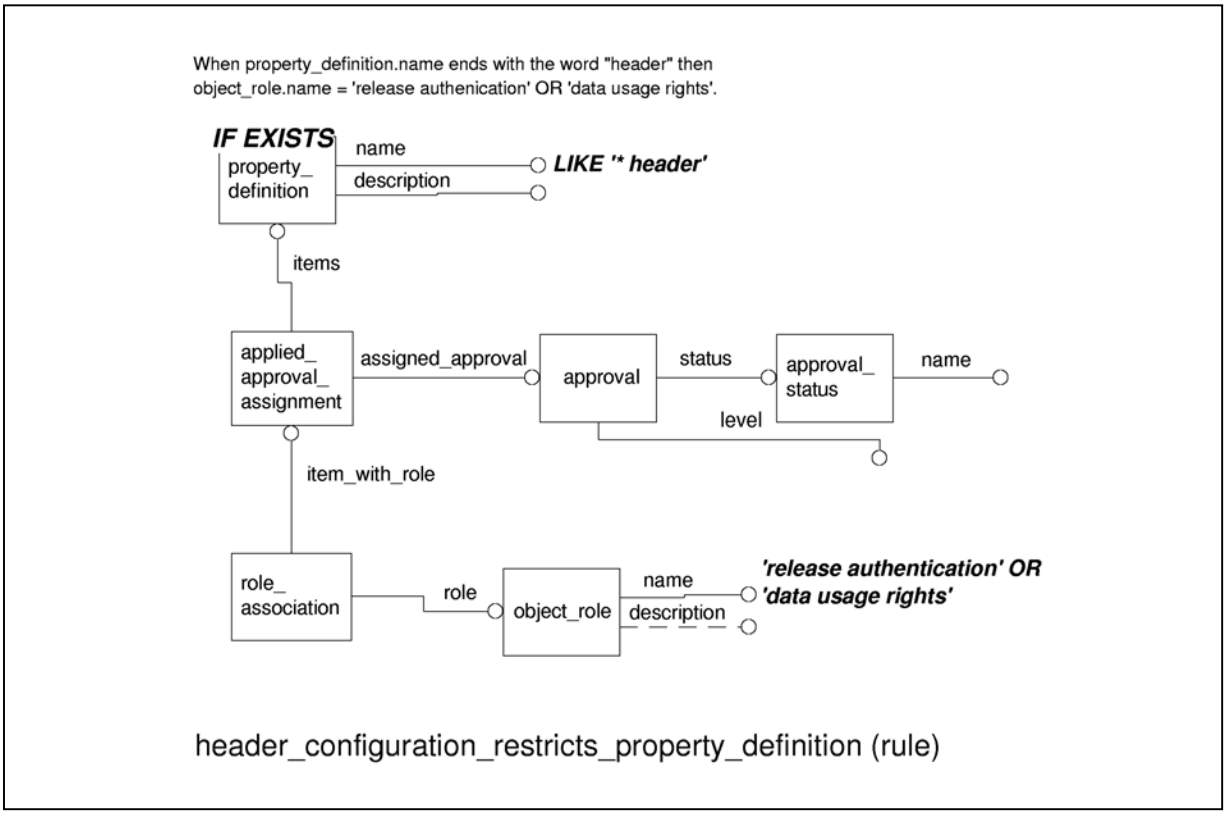


Figure 5.5 Rule Header configuration restricts property definition

5.5.7 Release authenication string restriction

The **release authentication string restrict** is not in the document but is suggested. It specifies how certain attribute string values in **approval, approval_role, identification_role, and approval_date_time_role** denote specific release authentication parameters.

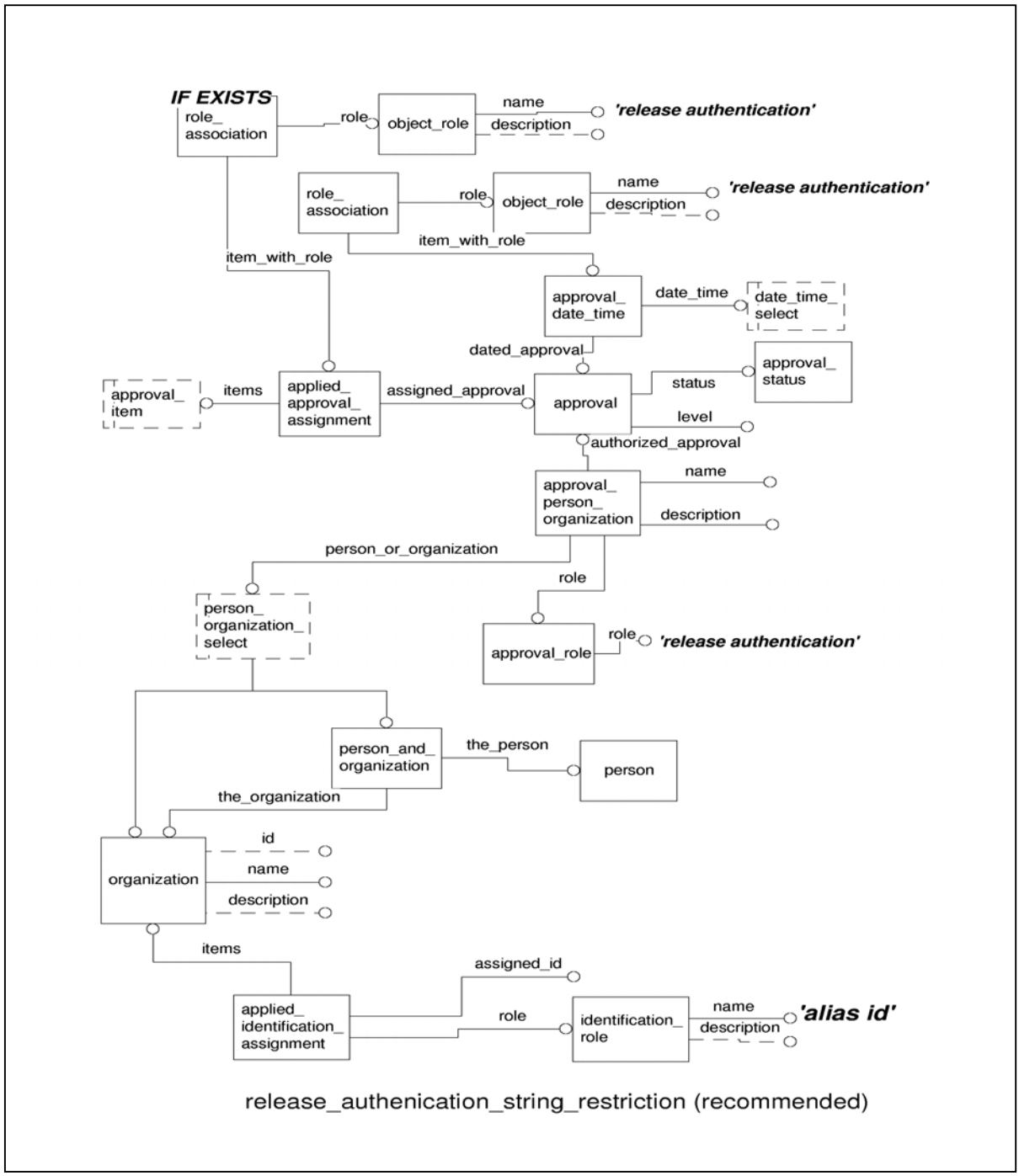


Figure 5.6 Suggested Rule Release authentication string restriction

5.5.8 Applied contract role constraint values

The applied contract role constraint values is not in the document but is suggested. It specifies the **object_role.name** attribute associated with **applied_contract_assignment** be constrained to identify the part or document the contract applies to or any data item description document related to the document.

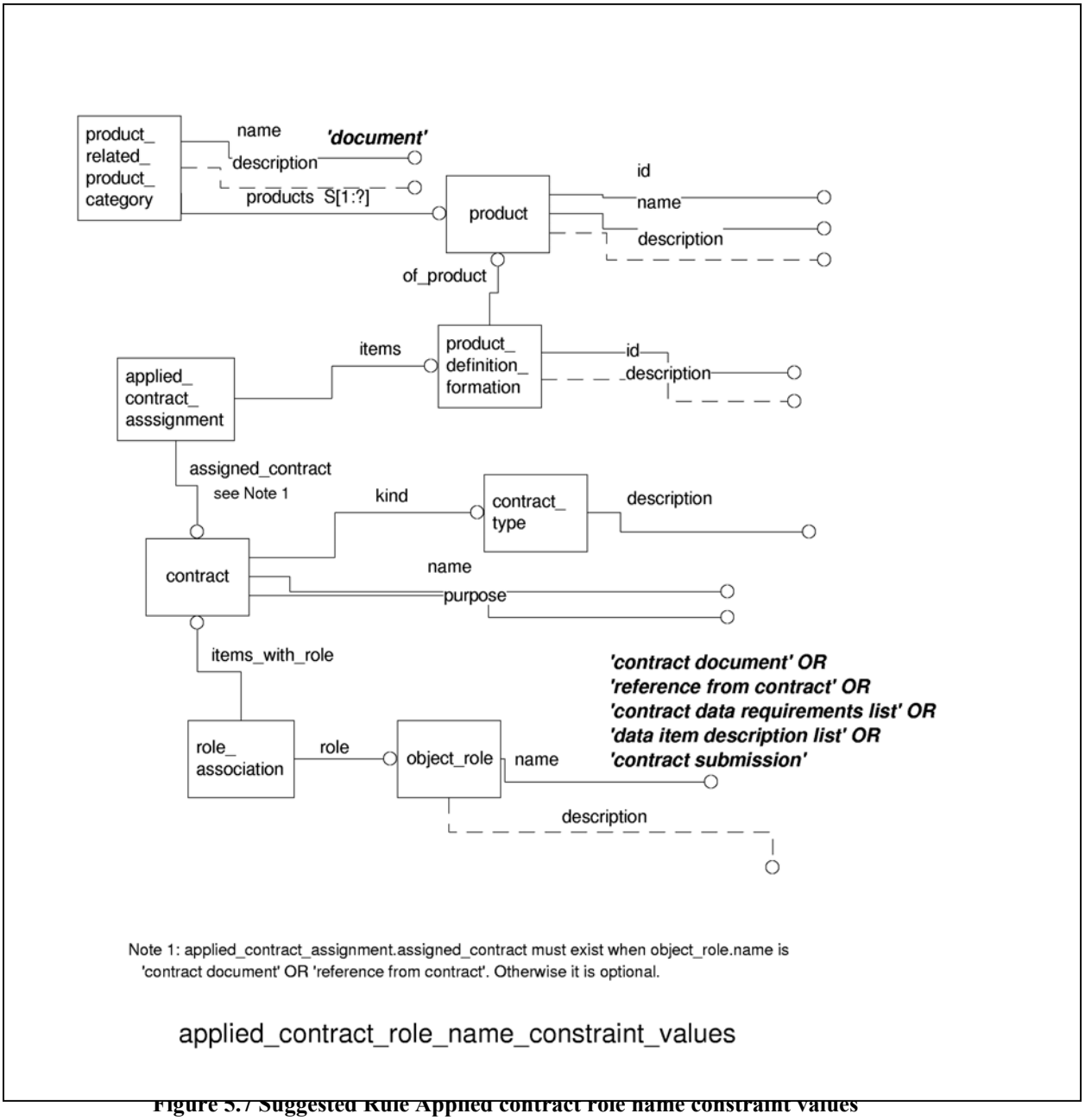


Figure 5.7 Suggested Rule Applied contract role name constraint values

5.5.9 Distribution notice requires supporting data

The **distribution notice requires supporting data.**

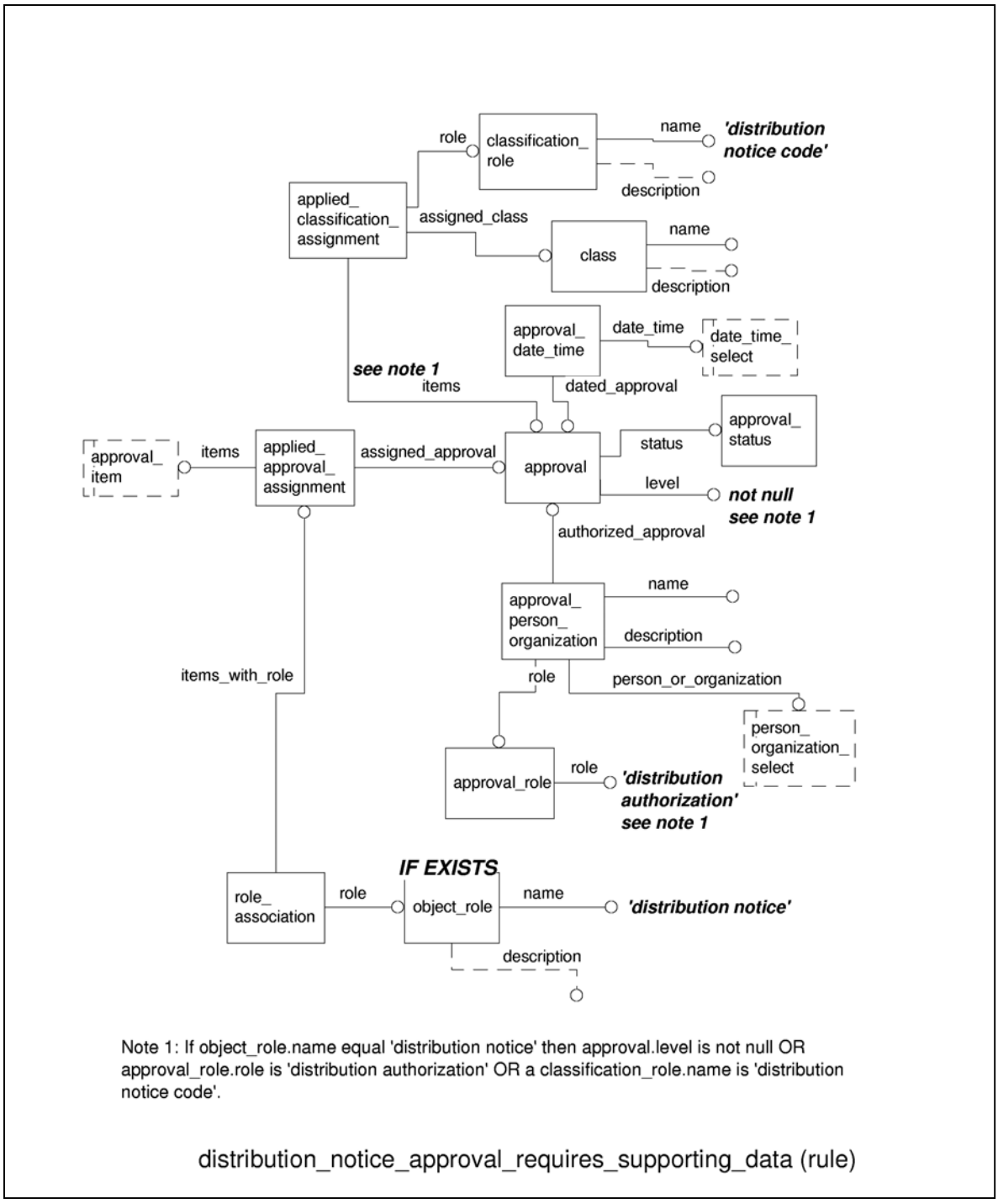


Figure 5.8 Rule Distribution notice approval requires supporting data

5.5.10 Security classification date string restriction

The **security classification date string restriction** rule specifies how certain attribute string values in **date_role and date_time_role** denote specific classification and declassification dates.

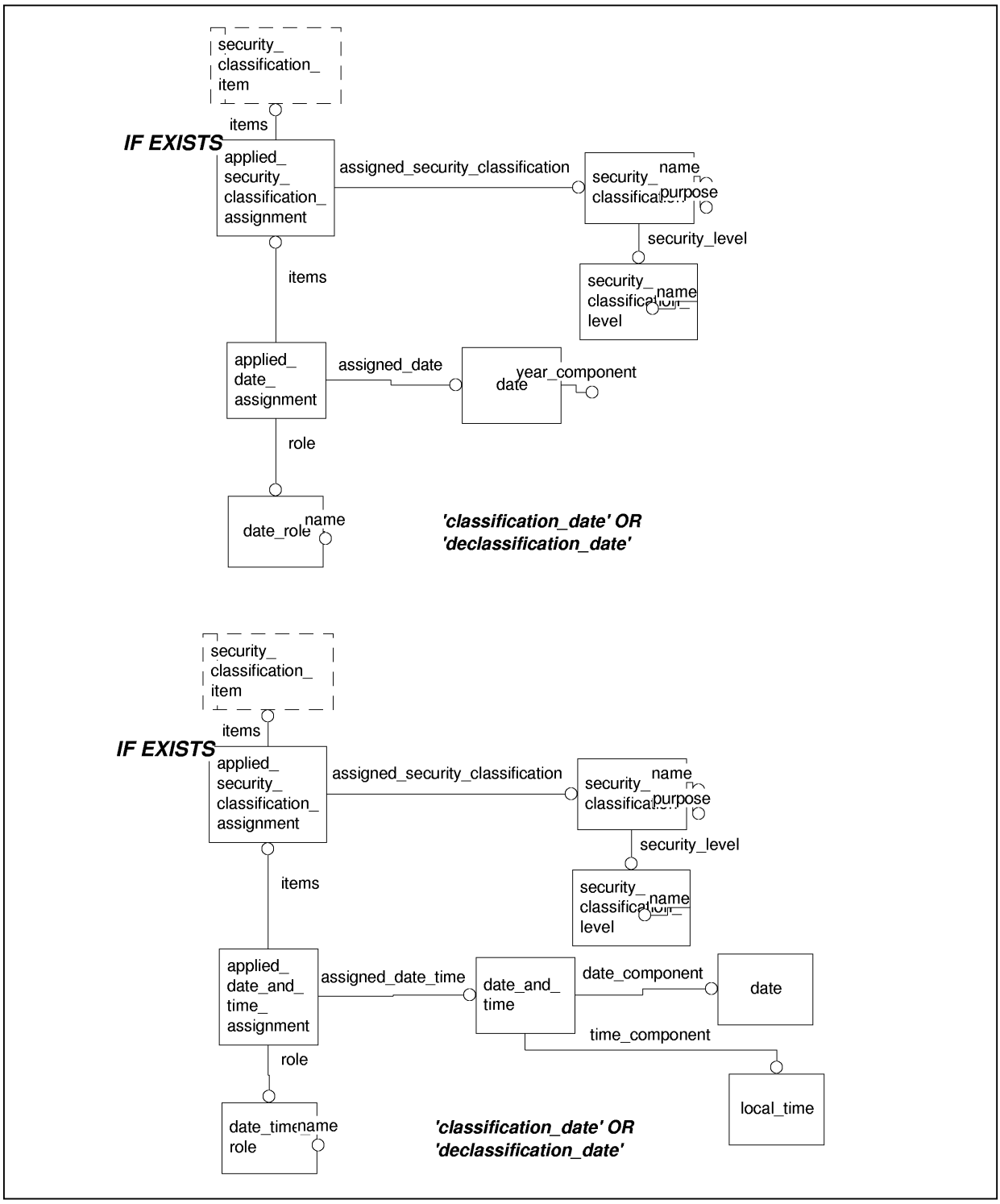


Figure 5.9 Rule security classification date string restriction

5.5.11 Status code basis identification constraint

The **Status code basis identification constraint** rule does not appear in the document but is suggested. It specifies the string values that are to be used when this optional attribue is present to identify status code groups and how to apply them.

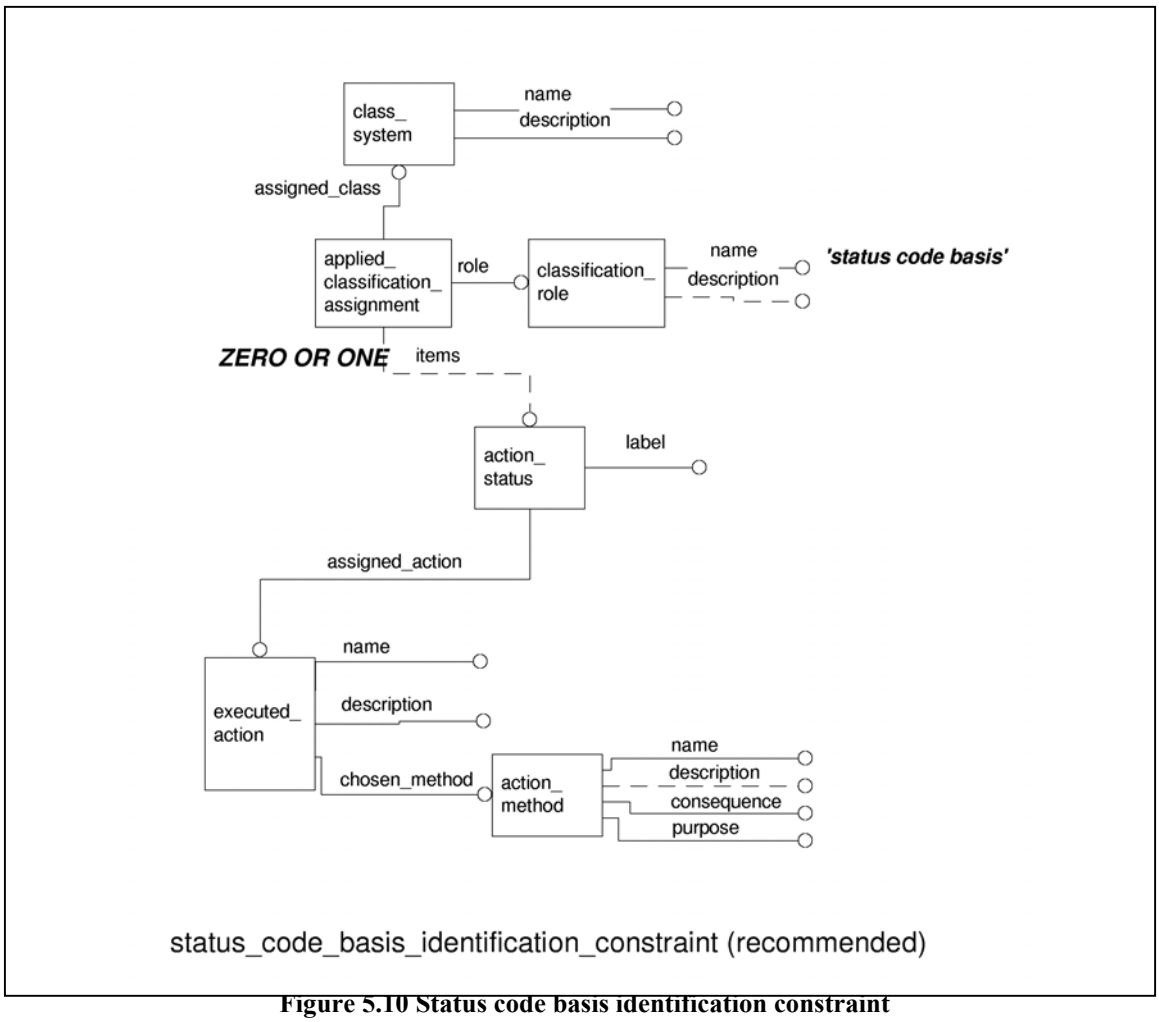


Figure 5.10 Status code basis identification constraint

5.5.12 Document requires subcategorization

The **document requires subcategorization** rule does not appear in the document but is suggested. It specifies that a reference document requires a lower level categorization using **product_-category.**

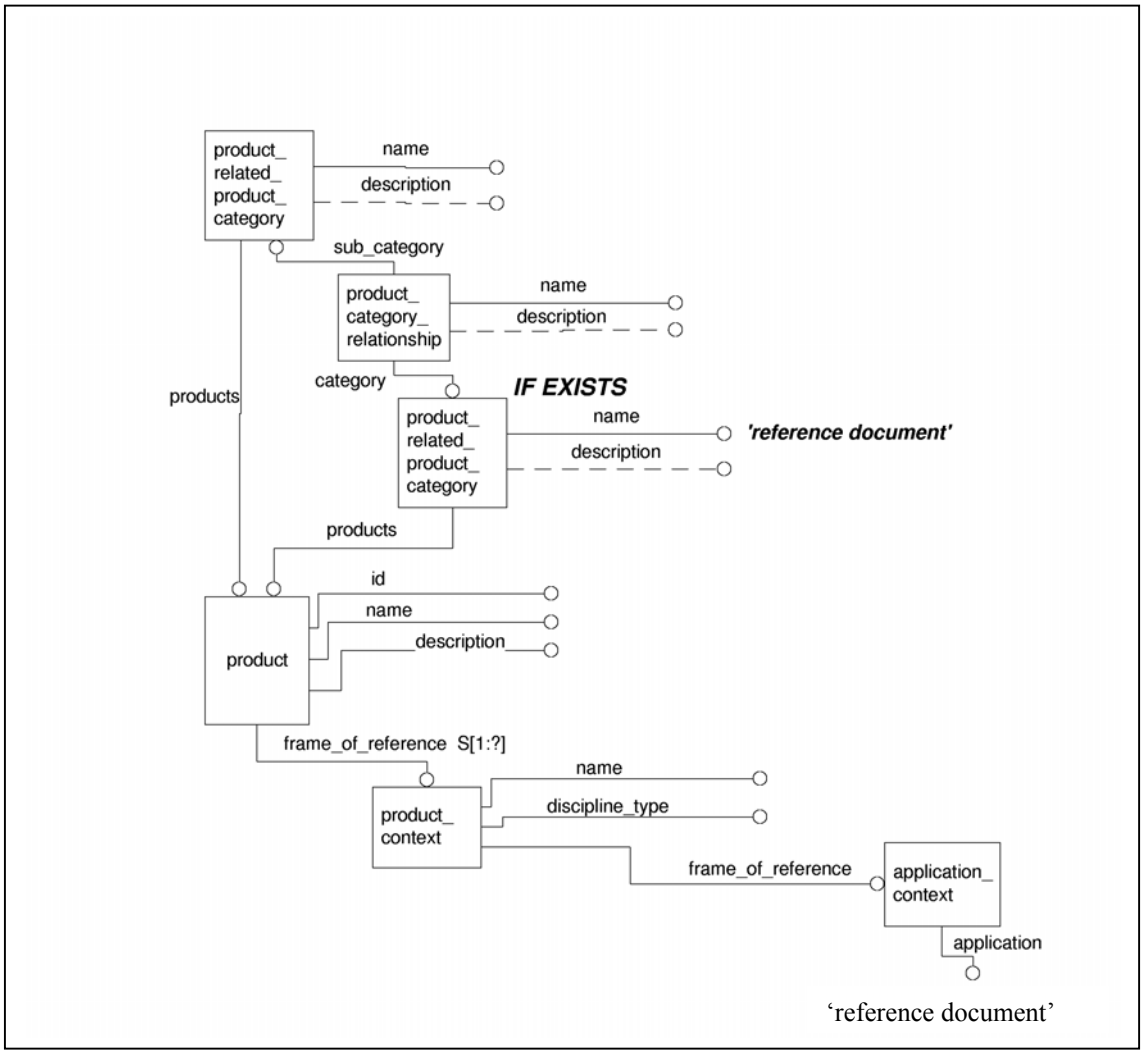


Figure 5.11 Rule Reference document requires subcategorization

6 Data Definition Exchange Modules

The data definition exchange modules are a collection of modules that, when used together, provide the necessary information needed to manage the exchange of information between two enterprises. These modules are used in conjunction with modules defined in the PDM schema. The recommended practices following are for the modules that are in this collection, but not in the core PDM schema recommended practice guide. There are supporting generic modules that are also needed and are described in sections 4 and 5. Section 6.1, 6.2, and 6.3 deal with three basic exchange levels, as described in the introduction, section 1.0.

The Data Definition Exchange (DDE) is made up of the first three conformance classes of AP 232. Each of these three conformance class provide a particular level of exchange management functionality. This section provides information about these three conformance classes in terms of proposed modules. Section 6.1 describes how AP 232 conformance class 3 provides exchange management utilizing a Product Structure Method. Section 6.2 describes how AP 232 conformance class 2 provides exchange managment utilizing a Document List Method. Section 6.3 describes how AP 232 conformance class 1 provides exchange management utilizing a File List Metho**d.**

6.1 Exchange Management Using Product Structure Method

This module could be called a top-level module because it pulls together all the other modules needed to make the equivalent of a conformance class within an Application Protocol. Exchange management using product structure exchange method is a type of document that is needed to manage the exchange of product data that satisfies a particular business purpose. The Exchange is managed by utilizing the product structure of parts and documents to maintain the context of the data being exchanged.

NOTE The structure exchange method facilitates the automated configuration control between parts, documents, and files.

Following is the high-level scope that this modules captures. Since this is the top-level module, many additional detail requirements are captured in lower level modules.

6.1.1 effective on

Specifies parameters that governs the use of the Item . The effective on need not be specified for a particular data definition indentured entry.

6.1.2 entry characteristics

Specifies the aspects about the indentured entry for the data definition indentured entry.

6.1.3 indentured entry

Specifies an Item_parent_to_item_child_relationship, a tdp_element_parent_to_item_child_-relationship, a Tdp_element_parent_to_tdp_element_child_relationship, Tdp_indentured_tdp_-element, an Item_parent_to_tdp_element_child_relationship, or a Tdp_indentured_item for the

Data_definition_indentured_entry and the relationship of the Tdp_element or item to other Tdp_elements or items.

Note 1 An Item_parent_to_item_child_relationship specifies that the row entry (child) is an Item and that the parent entry is an Item.

Note 2 A Tdp_element_parent_to_item_child_relationship specifies that the row entry (child) is an Item and that the parent entry is a TDP_element.

Note 3 A Tdp_element_parent_to_tdp_element_child_relationship specifies that the row entry (child) is a TDP_element and that the parent entry is a TDP_element.

Note 4 The Tdp_indentured_tdp_element specifies the top element of a Tdp_element based tree. This enables the listing of multiple document trees within the body of the list.

Note 5 An Item_parent_to_tdp_element_child_relationship specifies that the row entry (child) is a TDP_-element and that its parent entry is an Item.

Note 6 The Tdp_indentured_item specifies the top Items of an Item based tree. This enables the listing of multiple part-based document trees within the body of the list.

6.1.4 Indentured Level Notation

Indicates the number of indents for the Data_definition_indentured_entry from the Item or Tdp_-element identified in the indentured_entry. The level of indenture indicates a relative position between the Data_definition_entry and the top level item, or Tdp_element that is specified in the indentured_entry. The indenture_level need not be specified for a particular Data_definition_-entry.

Note 1 The level of indenture is formulated based on the document tree or the item tree from the top level Tdp_element or item identified in the data_definition_indentured_entry.indentured_entry.

Note 2 Industrial practices have the TDP_element or Item as the first level of indenture, while the TDP_-elements or Items that are referenced from the top level TDP_element or Item are at the second level of indenture. Lower levels of indenture are specified relative to the top level TDP_element or Item.

6.1.5 Document Header

The document header specifies sufficient configuration control and data management information for a specific Data definition exchange

6.1.6 Revision History

The revision history specifies a record of revisions of the Data definition exchange using product structure exchange_method. The revision history shall be constructed in a manner whereby the first entry in the tabulation is the first revision, the second entry is the second revision, and subsequent entries in the tabulation are subsequent revisions. The revision_history shall not contain an entry for the original release. The revision history need not be specified for a particular Data_definition_exchange_using_product_structure_exchange_method. There may be more than

one revision_history for a Data_definition_exchange_using_product_structure_exchange_-method.

6.1.7 Notes List

The notes list specifies notes that are applicable to the data definition exchange using product structure exchange method. Some notes are referenced by the use of a note reference code in any section of the data defintion exchange using product structure exchange_method. The notes describe any explanatory remarks or notations that may be useful or informative regarding the subject that referenced the note. The notes_list need not be specified for a particular data definition exchange using product structure exchange method. There may be more than one notes list for a Exchange management using product structure exchange method.

6.1.8 Method of Entry Tabulation

The method of entry tabulation specifies an Indentured list by part, an indentured list by document, or an indentured list by part with document references to parts that specify how the entries in the order_of are specified for the data definition exchange using product structure exchange method, or indentured data definition list.

NOTE The three methods are defined by the indentured list by part, the indentured list by part with document references to parts, or the Indentured list by document.

6.1.9 Order of Indentured Entries

The order of specifies the list of Data definition indentured entry(s) for the data definition exchange using product structure exchange method. The order in the listing is specified by the method of entry tabulation. There may be more than one order of for a data definition indentured list method,

6.1.10 AIM Entity and Attributes

The instance diagrams, Figure 6.1 and Figure 6.3, show the entities that this module either defines or constrains. These entities establish the framework for tying all the information together for this exchange management document. The italicized text in the diagrams represents the requirements of the module. The box arrows in the diagrams point to where the requirement is satisfied. The diagram's attributes that are not shown will follow the rules established by lower level modules. The modules that make up the PDM schema are considered lower level modules.

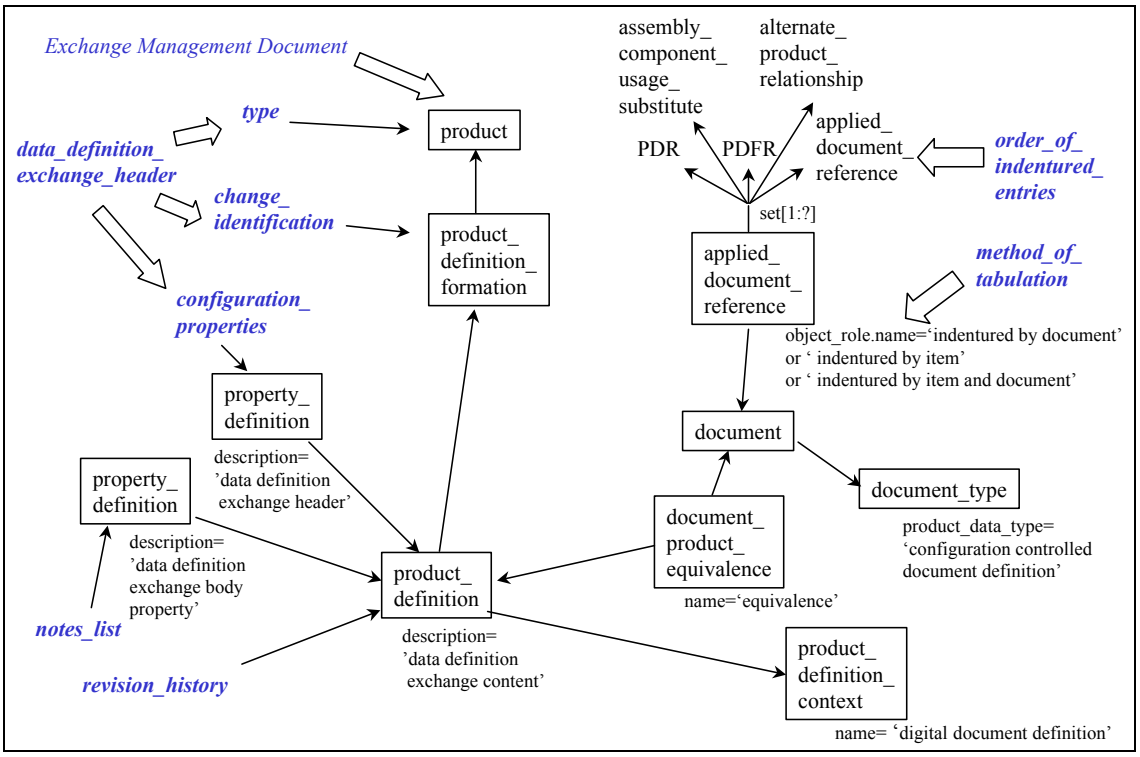


Figure 6.1 - Exchange Management Using Product Structure Exchange Method

Figure 6.1 shows how a **product_definition** is established to gather the content information of the exchange management document. In order to collect the parts and documents that are to be exchanged, the **product_definition** is associated with a document construct, which then utilizes an **applied_-document_reference** to do the collecting.

Figure 6.2 shows how the collection of product data is identified through the use of an indentured list method. The key feature here is that the documents and parts are identified through a particular usage. This usage is represented by entities that associate parts and documents together. These associations establish the parent-child relationship required for the positioning elements within the indentured list.

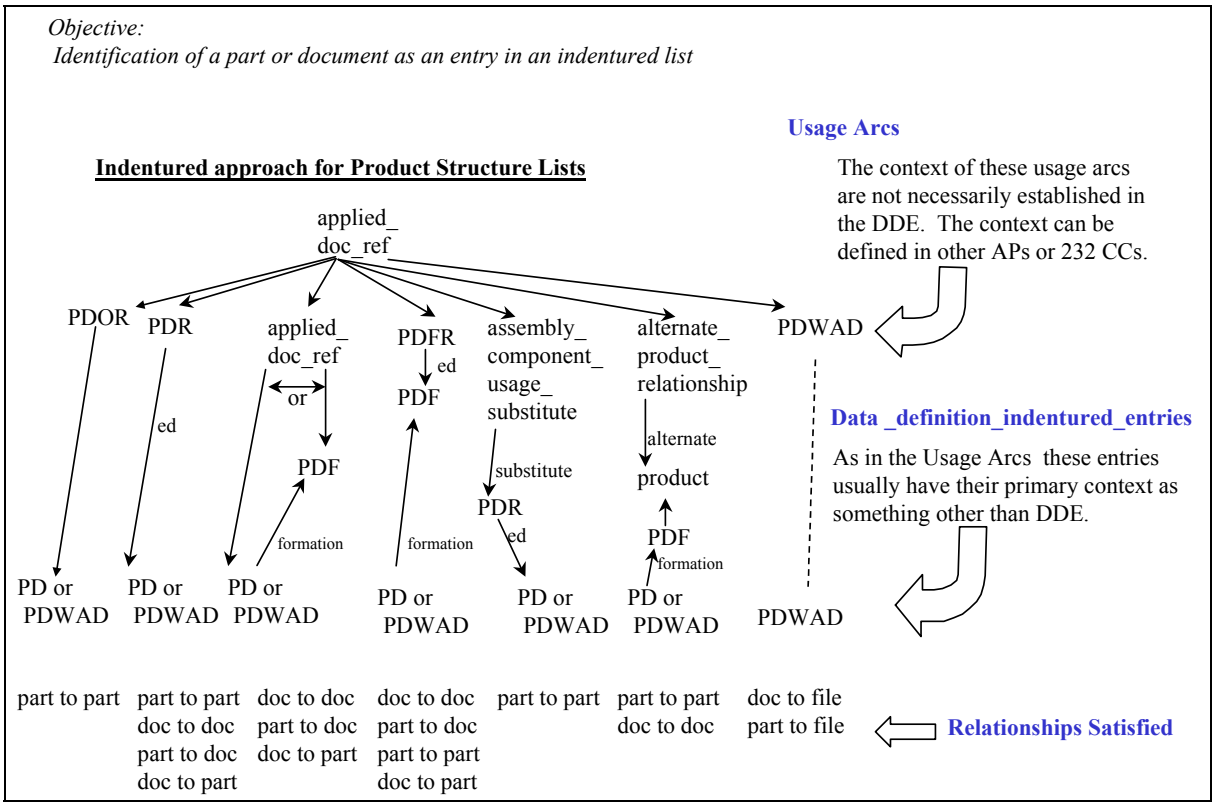


Figure 6.2 - Exchange Management Usage Arcs

Figure 6.3 shows two unique indentured list entry parameters: effective on and indentured level - notation. The effective on parameter points to the effectivity module. The indentured –level notation uses a **descriptive_representation_item.name** to capture a text notation for a particular indenture level. This can be a number (e.g., 1, 2, …) or an alpha character (e.g., A, B, C,…Z, AA . . . ZZZ). With or without the indentured_level_notation being instantiated, the actual indentured level can be derived from the indentured list.

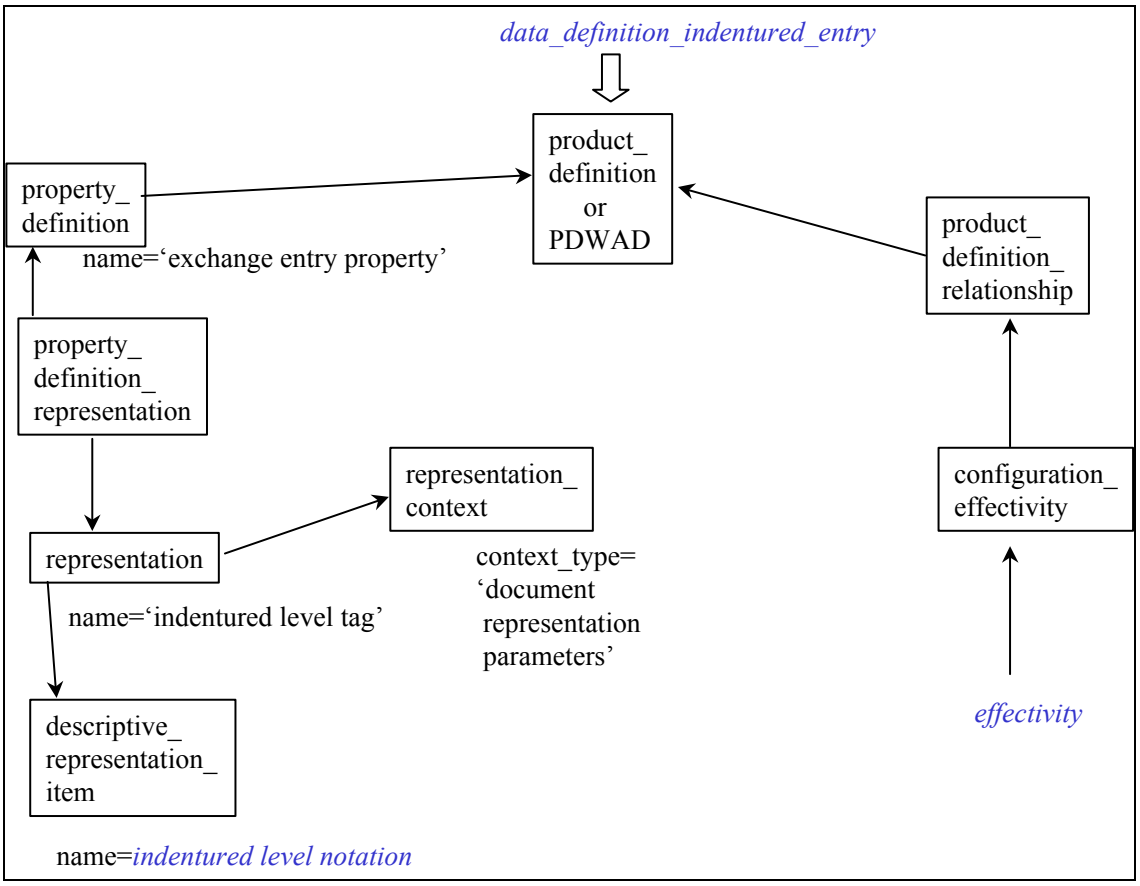


Figure 6.3 - Exchange Management entry indentured level and effective on

6.2 Exchange Management Using Document List Method

This module could be called a top-level module because it pulls together all the other modules needed to make the equivalent of a conformance class within an Application Protocol. Exchange Management Using Document List Method is a type of document that can manage the exchange of product data that satisfies a particular business purpose. The Exchange consists of identifying a set of individual documents with or without their representative files. The exchange management using document list module has all the same basic components as the Exchange Management Using Product Structure Method module, except for the indentured method that is replaced by a set of documents. Following is the high-level scope that this module captures. Because this is the top-level module, many additional detail requirements are captured in lower level modules.

6.2.1 AIM Entity and Attributes

Figure 6.4 shows how a **product_definition** is established to gather the content information of the exchange management document. In order to collect the documents that are to be exchanged, the **product_definition** is associated with a document construct, which then utilizes an **applied_-document_reference** to do the collecting. Figure 6.4 is almost the same as Figure 6.1, except for some string constraints and that the **applied_document_reference** only points to the **product_-**

definition views (**product_definition** or **product_definition_with_associated_documents**) of documents.

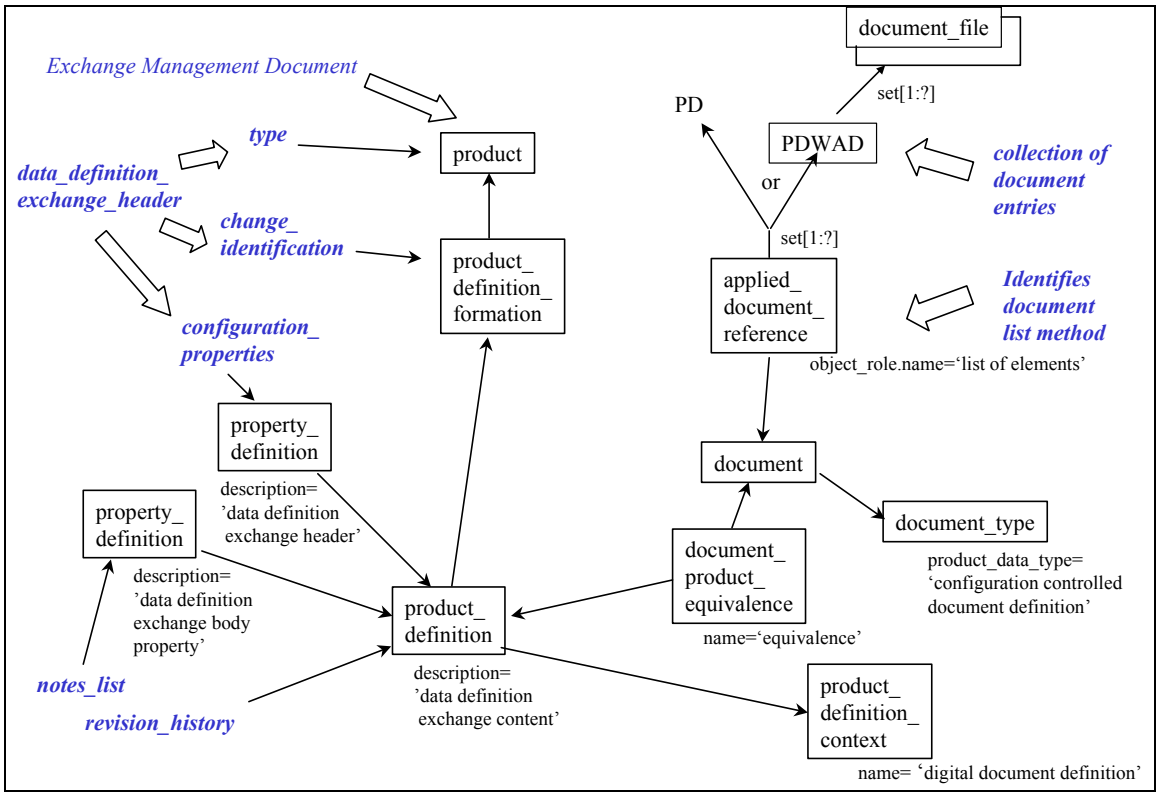


Figure 6.4 - Exchange Management Using Document List Method

6.3 Exchange Management Using File List Method

This module could be called a top-level module because it pulls together all the other modules needed to make the equivalent of a conformance class within an Application Protocol. Exchange Management using file list method is a type of document that can manage the exchange of product data that satisfies a particular business purpose. The exchange consists of identifying a set of individual files. The exchange management using file list module has all the same basic components as the data definition exchange using document list module, except for the list of documents which is replaced by just a set of files. Following is the high-level scope that this module captures. Since this is the top-level module many additional detail requirements are captured in lower level modules.

6.3.1 AIM Entity and Attributes

Figure 6.5 - Exchange Management Using File List Method
shows how a **product_definition** is established to gather the content information of the exchange management document. In order to collect the files that are to be exchanged, the product_definition is associated with a document construct, which then utilizes an **applied_-document_reference** to do the collecting. Figure 6.5 - Exchange Management Using File List Method
is almost the same as Figure 6.1 except for some string constraints and that the **applied_-document_reference** only points to files (document_file).

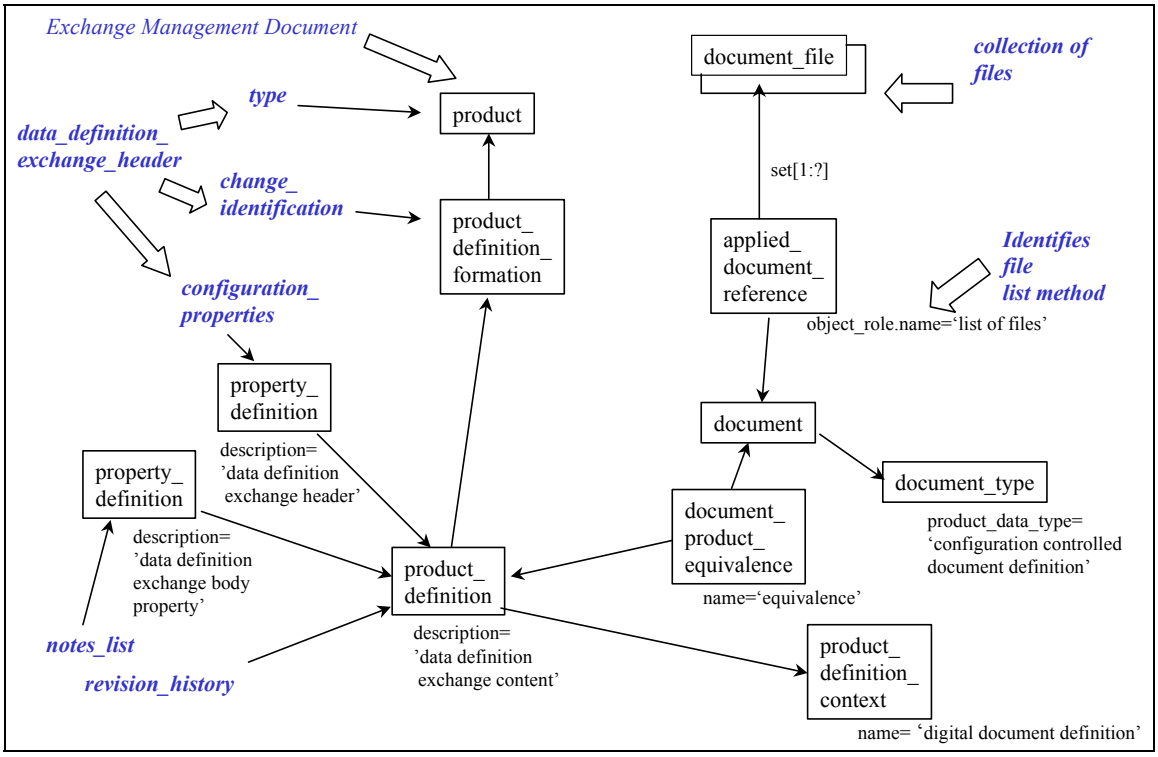


Figure 6.5 - Exchange Management Using File List Method

6.4 Indentured List Method

The indentured list method application module supports the requirement to specify indentured lists of product information. The product information, which is the subject of the indentured lists, is a product (parts and documents). The indentured list method utilizes a parent-child paradigm to identify placement order in the indentured lists. A collection of parent-child relationships of parts and documents make up an indentured list. This application module defines three indentured list method types. The three indentured list method types are Indentured_list_by_-part, Indentured_list_by_document, and Indentured_list_by_part_with_document_references_-to_parts. Each of the three methods specify which types of relationships (part to part, part to document, document to document, document to part) may be utilized to identify entries in a particular indentured list.

NOTE There are four types of parent-child relationships that are utilized to formulate the three types of indenture lists. The Table 1 below shows which parent-child relationships are used in the three types of indentured lists.

Table 1 - Parent-child relationships used in indenture list types

INDENTURED LIST METHOD TYPE	PARENT-CHILD			
	PART/PART (Item/Item)	PART/DOCUMENT (Item/TDP_element)	DOCUMENT/PART (TDP_element/ Item)	DOCUMENT/DOCUMENT (TDP_element/TDP_element)
Indentured by Document				YES *
Indentured by Part	YES *	YES		YES
Indentured by Part (with Document references to Parts)	YES *	YES *	YES *	YES *

NOTE (*) -- Means Allowable Top Indenture

Indenture by Document- Provides the ability to capture the relationships of documents (Tdp_elements) within the exchange. A hierarchical indenture list that provides a parent-child relationship between the Tdp_elements is supported. Applications can query this list to identify what documents are referenced from any other. For example, the relationship between an assembly drawing and its subordinate documents (detail drawings, material specification, etc.) may be captured.

This level of capability allows for the capture and exchange of the following information:

— identifies parent-child relationships between Tdp_elements;

— allows for a hierarchical order of indenture to be captured among Tdp_elements.

Indenture by part - Provides the ability to capture the relationships of parts within the exchange. A hierarchical indentured list that provides a parent-child relationship between parts and assemblies, parts and their Tdp_elements, and Tdp_elements and their subordinate Tdp_elements is supported. Applications may query this list to identify what documents are available about a part, which parts go into which assembly, and which documents are referenced from other documents. For example, utilizing the Indenture by Part capability level, the drawing a part is defined in and the assembly that a part is a portion of could be determined by knowing the identification of a part.

This level of capability allows for the capture and exchange of the following information:

— identifies parent-child relationships between parts;

— identifies parent-child relationships between a part (parent) and a Tdp_element (child);

— identifies parent-child relationships between Tdp_elements;

— allows for a hierarchical order of indenture to be captured among parts and Tdp_elements with the part always playing the higher indenture role when associated directly with a Tdp_element;

— identifies that a part always occupies the first indenture level.

Indenture by part (with document references to parts) - Provides the ability to capture the relationships of parts and/or Tdp_elements within the exchange. A hierarchical indentured list that provides a parent-child relationship between parts and assemblies, parts and their Tdp_elements, Tdp_elements and their parts, and Tdp_elements and their subordinate Tdp_elements is supported. Applications may query this list to identify what documents are available about a part, what parts go into which assembly, what parts are defined in a drawing, and what documents are referenced from other documents. For example, utilizing the Indenture by Part (with Document References to Parts) capability level, the parts defined on an assembly drawing could be identified by just knowing the identification of the assembly drawing.

This level of capability allows for the capture and exchange of the following information:

— identifies an implied parent-child relationships between parts;

— identifies parent-child relationships between a part (parent) and a Tdp_element (child);

— identifies parent-child relationships between a Tdp_element (parent) and a part (child);

— identifies parent-child relationships between Tdp_elements;

— allows for a hierarchical order of indenture to be captured among parts and Tdp_elements with either the part or the Tdp_element playing the higher indenture role;

— allows that either the TDP_element or part may occupy the first indenture level.

6.4.1 AIM specification

The indentured list method is mapped to the integrated resources as a uniquely identified **applied_document_reference** which points to a collection of Relationship /Association Entities that establish the parent-child relationships for Indentured List's nodes. The **applied_-document_reference** is uniquely identified by its associate **object_role.name** The valid values of **object_role.name** are 'indentured by document', 'indentured by item', and 'indentured by item and document'. These three values correspond to the three indentured list method types describe above. The valid parent-child relationships that can be used in a specific indentured list are based on which indentured list method is selected. Table 1 shows when each type of parent-child relationship may be used.

There are a variety of parent-child relationship entities that can be used to establish the chaining of nodes within the indentured list. These parent-child relationship entities are shown in Figure 6.6 – Indentured Approach for Product Data Structure
. Some of the parent-child relationship entities on are supertypes of other entities. The subtypes of these supertype parent-child relationship entities are valid to use also. Figure 6.6 – Indentured Approach for Product Data Structure
also shows the paths between the parents and children.

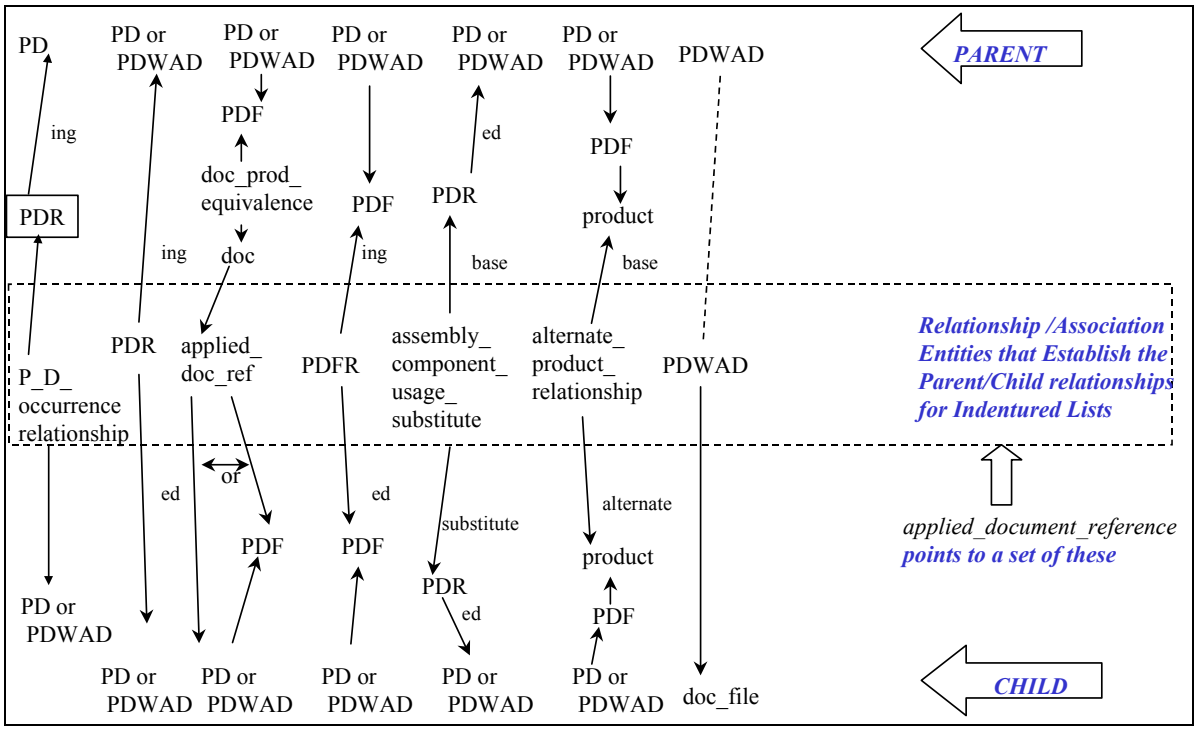


Figure 6.6 – Indentured Approach for Product Data Structure

The purpose of identifying the arcs instead of the nodes is to ensure that the correct usage path is identified. Figure 6.7 shows a product/document tree structure. In this tree structure the relationship (arcs) entities have boxes around them. Entities in boxes, in Figure 6.7, are pointed to by applied_document_reference, which acts as the collector for the indentured list. PDWAD plays two roles - it serves as an identifier or node of the document or a part in the tree structure; and it serves as an arc that relates the document or part to representative files. The italicized text are nodes in the tree structure.

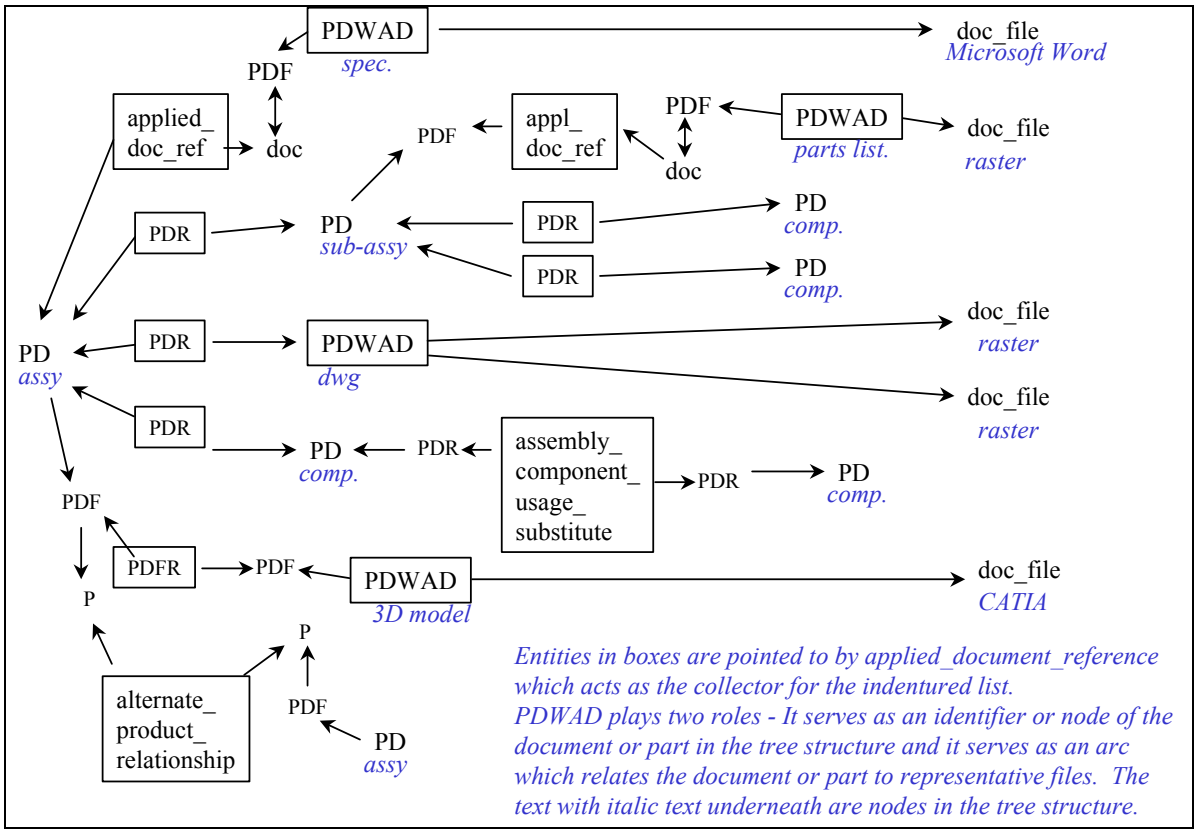


Figure 6.7 - Example Indentured by Part for Exchange Management

As a portion of the SET of arcs (relationship entities) pointed to by the **applied_document_-reference**, the top-level node can also be included from the indentured list. This is allowed is to aid in processing, and is especially true when there are multiple indentured lists combined together and managed in one exchange. The **applied_document_reference** entity would point to arcs from an indentured list and its top node. This is shown in Figure 6.8 - Top Nodes Identified in List for Exchange Management

.

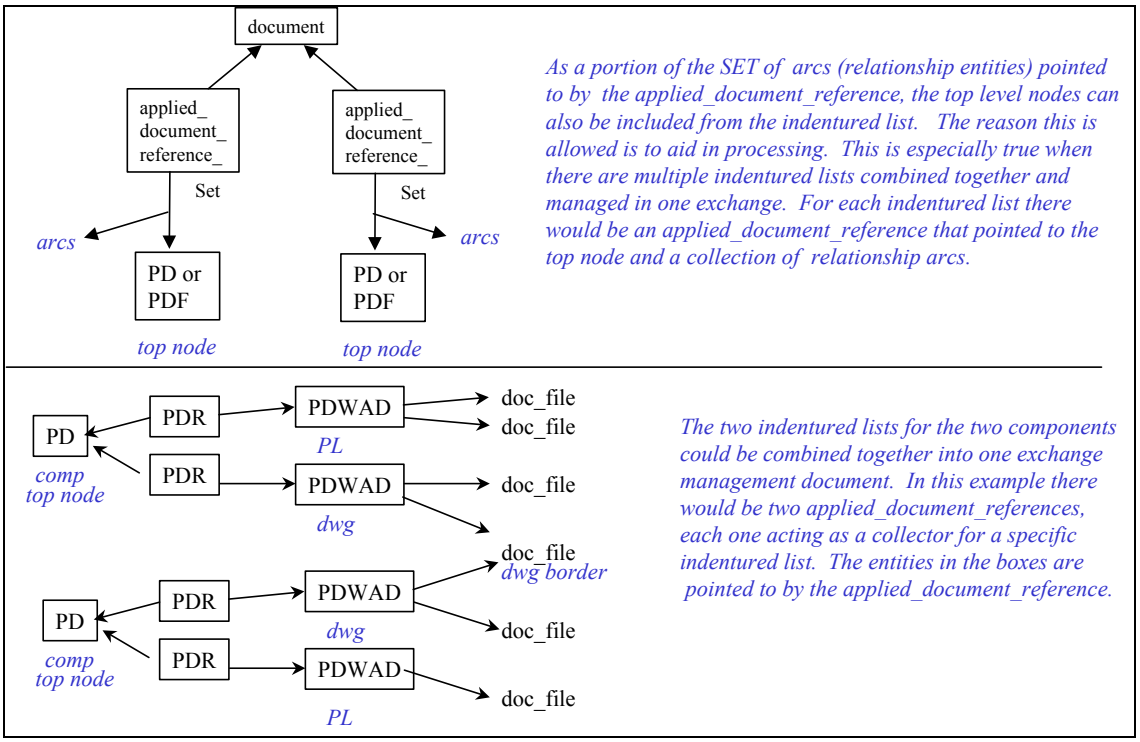


Figure 6.8 - Top Nodes Identified in List for Exchange Management

6.5 Data Definition Entry

The data definition entry is a collection of characteristics about an item (part), or Tdp_element (document) in a data exchange. The collection of characteristics are typical characteristics about the data that is maintained within Product Data Management (PDM) or configuration control systems. The characteristics are employed to manage the use and distribution of the File, Item, or Tdp_element.

The data definition entry module is used in both the exchange management using product structure method and the exchange management using document list method modules.

6.5.1 AIM Specification

A **product_definition** or **product_definition_with_associated_documents** represents a Data_-definition_entry. This **product_definition** can be for a part or a document.

When the **product_definition** is for a document there are multiple possible **application_contexts** that could exist. These **application_contexts** are described in document_definition module and document_header module. The possible values for **application_context.name** are ‘document definition’, ‘digital document occurrence’, and ‘physical document occurrence’.

When the **product_definition** is for a part, part_definition defines the **application_contexts.**

Figure 6.9 - Data Definition Entry Characteristics

provides the basic paths between the data definition entry (**product_definition**) and each of the data definition entry's characteristics. There are twelve characteristic properties that can be applied to the data definition entry. Five of the characteristics utilize the same occurrence of **property_definition** in their path. The twelve characteristics are the following:

— Include Flag;

— Master Representation Flag;

— Special Condition;

— Notation;

— Data Usage Rights;

— Available From – Contract Submission;

— Available From – Document Reference;

— Change Information – Superceded

— Change Information – Version of the Exchange Management Document a Entry Applies;

— Associated Files;

— Document Format and Size;

— Delivery Accounting Reference Documents.

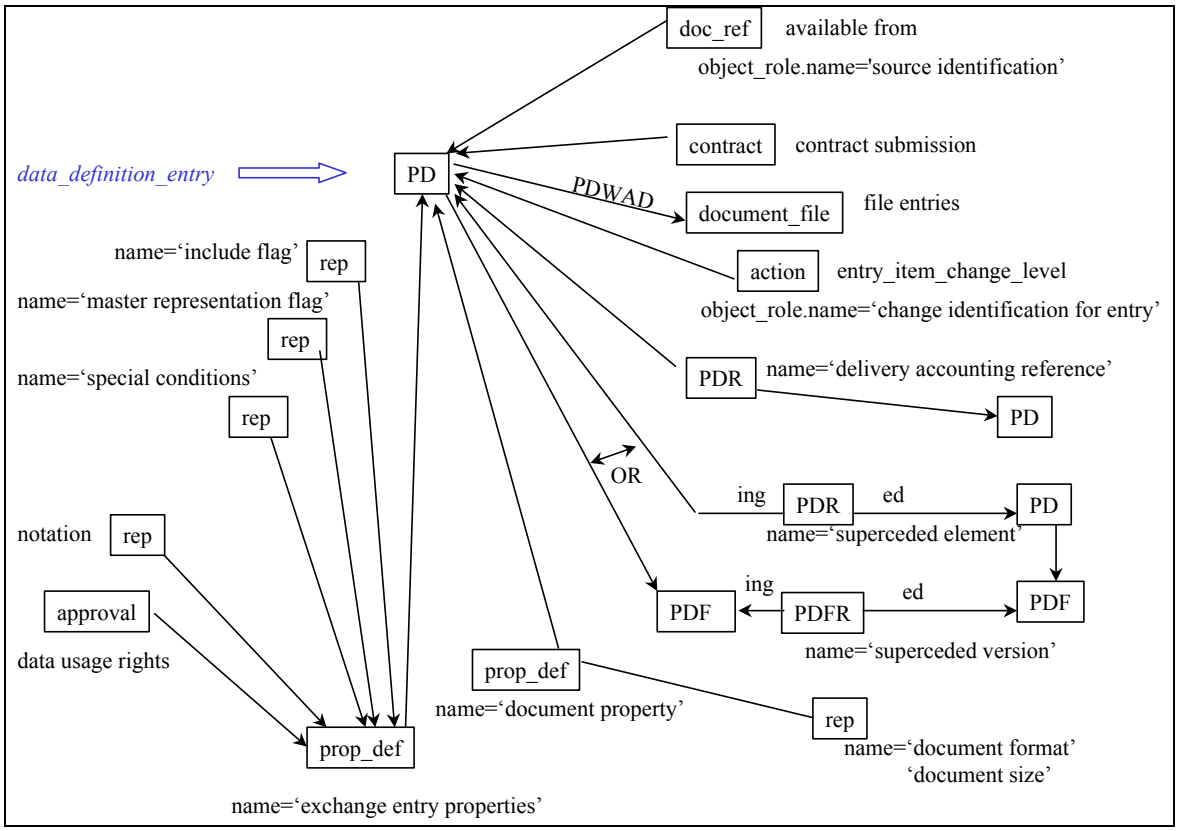


Figure 6.9 - Data Definition Entry Characteristics

6.5.1.1 Include Flag

The include flag characteristic is shown in Figure 6.10 – Entry Characteristics (Include Flag, Master Rep. Flag, Special Conditions)

This specifies whether or not the file, item (part), or Tdp_element (document) is included or furnished as a portion of this exchange. The include flag information is captured in a **descriptive_representation_item**. The valid values for **descriptive_representation_item.name** are 'already delivered', 'available for access', 'fully included', 'not deliverable', 'not delivered yet', 'not received', 'partially included', 'receipt accepted', 'receipt acknowledged', 'receipt partially rejected', 'receipt rejected', and 'resubmission'. This list of values is intended to cover a wide range of exchange status include flags. In the event that this enumeration of include flags does not support a required exchange status, business exchange partners can add to this list and these additions should be added to this recommended practice usage guide. The representation occurrence that points to the **descriptive_representation_item** shall have its name attribute contain the string 'include flag'. The associated **representation_context** occurrence will have its name attribute contain the string 'document parameter'. The **property_definition_-representation** occurrence that connects the **representation** and the **property_definition** together will have its string attributes be '/Null' or a blank string''. The **property_definition** occurrence will have its name attribute contain the string 'exchange entry properties'.

Table 2 shows these string constraints and where they apply.

The meaning for each include flag is the following:

6.5.1.1.1 already delivered
already delivered specifies that the Tdp_element data is not in the exchange because it was exchanged previously;

6.5.1.1.2 available for access
available for access specifies that the Tdp_element data is available for the receiving system to remotely access the data. The Tdp_element data is not in the exchange;

6.5.1.1.3 fully included
fully included specifies that the Tdp_element data and all of the data files are in the exchange;

6.5.1.1.4 not deliverable
not deliverable specifies that the Tdp_element data are not in the exchange. The Tdp_element data are either not deliverable or planned for delivery;

6.5.1.1.5 not delivered yet
not delivered yet specifies that the Tdp_element data are not in the current exchange. The data will be exchanged in a future exchange;

6.5.1.1.6 not received
not received specifies that the Tdp_element data were not received in a referenced exchange or shipment;

6.5.1.1.7 partially included
partially included specifies that components of the Tdp_element data are exchanged. Some of the data files comprising the Tdp_element are exchanged;

6.5.1.1.8 receipt_accepted
receipt accepted specifies that the Tdp_element data received in the referenced exchange were successfully received and accepted as approved by the receiver;

6.5.1.1.9 receipt acknowledgement
receipt acknowledgement specifies that the Tdp_element data received in the referenced exchange or shipment were successfully received;

NOTE The receipt acknowledgement does not imply acceptance by the receiver

6.5.1.1.10 receipt partially rejected
receipt partially rejected specifies that the Tdp_element data received in the referenced exchange or shipment were partially rejected due to non-compliance with exchange or delivery requirements;

6.5.1.1.11 receipt rejected
receipt rejected specifies that the Tdp_element data received in the referenced exchange or shipment were rejected due to non-compliance with the exchange or delivery;

6.5.1.1.12 resubmission
resubmission specifies that the Tdp_element data are in the exchange as a resubmission or resend.

Table 2 - string constraints

Property_-definition.name	**Representation .name**	**Representation_ context .context_type**	**Descriptive_ representation_item .name**	**Measure_ Representation .name**
‘exchange entry properties’	‘include flag’	‘document representation parameters’	‘already available’	
“	“	“	‘available for access’	
“	“	“	‘fully included’	
“	“	“	‘not deliverable’	
“	“	“	‘not delivered yet’	
“	“	“	‘not received’	
“	“	“	‘partially included’	
“	“	“	‘receipt accepted’	
“	“	“	‘receipt acknowledged’	
“	“	“	‘receipt partially rejected’	
“	“	“	‘receipt rejected’	
“	“	“	‘resubmission’	
“	‘special condition’	*type of coding schema*	*special condition code*	
“	‘notation’	‘document representation parameters’	*reference code*	
“	‘master representation flag’	‘document representation parameters’	‘master representation’	

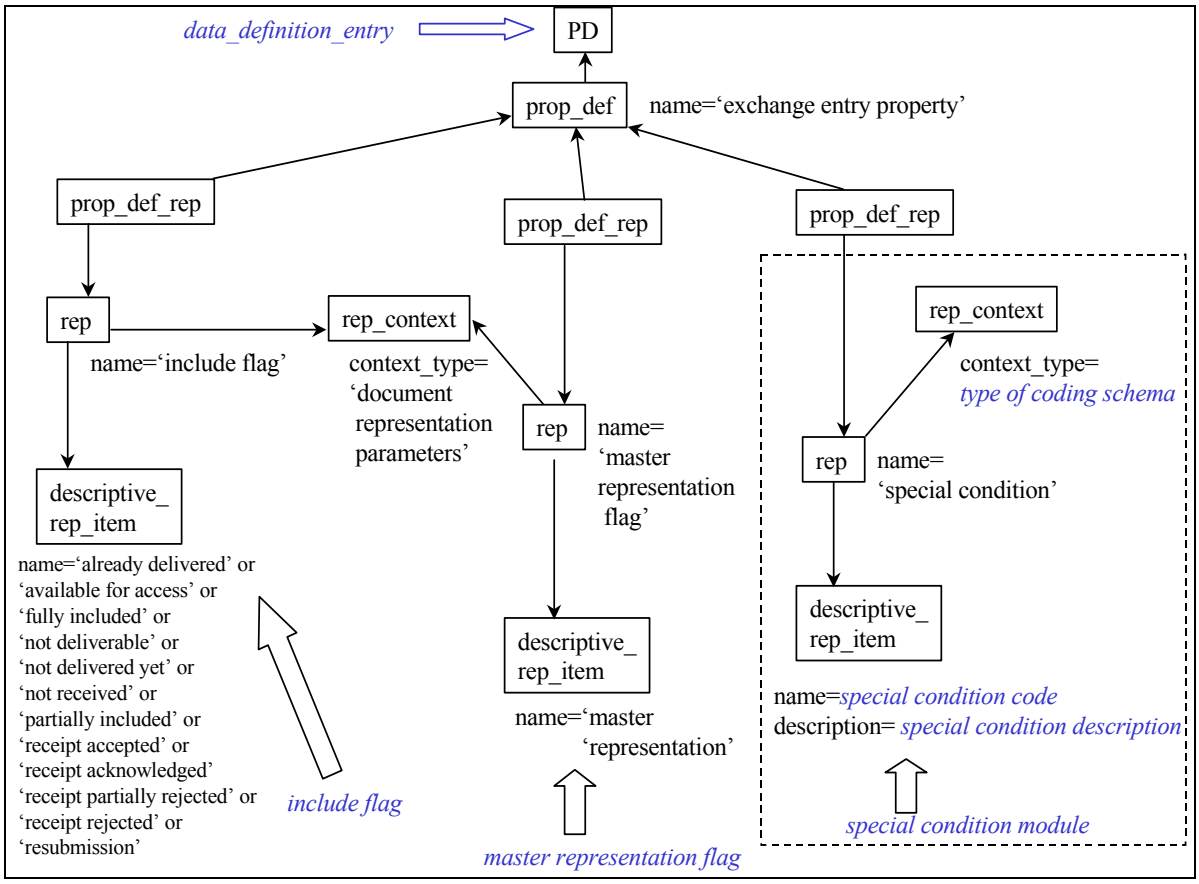


Figure 6.10 – Entry Characteristics (Include Flag, Master Rep. Flag, Special Conditions)

6.5.1.2 Master Representation Flag

The master representation flag identifies this representation as the master, shown in Figure 6.10 – Entry Characteristics (Include Flag, Master Rep. Flag, Special Conditions)
. This means that all other representations of this particular document version originated from this **representation**. The unique string constraints of this characteristic are **representation.name**='master representation flag' and **descriptive_representation_-item.name**='master representation'. These two constraints with this characteristic's other constraints are shown in Table 2

6.5.1.3 Special Condition

Special conditions placed on the entry are shown in Figure 6.10 – Entry Characteristics (Include Flag, Master Rep. Flag, Special Conditions)
. This special condition is a characteristic of an Item or Tdp_element that is mutually agreed to among parties for a business purpose. There are three pieces of information that make up this characteristic. The first is a code that specifies a specific identifier from a set of mutually agreeable special conditions. The string of characters that make up the code is captured in the **descriptive_representation_item.name** attribute. The second is a description of the special condition. The description is captured in the **descriptive_representation_item.description** attribute. Both the code and description information are optional but one must exist. The third is

a type of coding schema. Coding schemes can be established by companies and industries that agree on different codes to depict different conditions. These coding scheme agreements are sometimes contained in industry or government standards. An example of a United States standard that contains a special_condition coding scheme is ANSI ASME Y14.34M. The coding scheme is captured in the **representation_context.context_type** attribute. Constrained string values are shown in Table 2

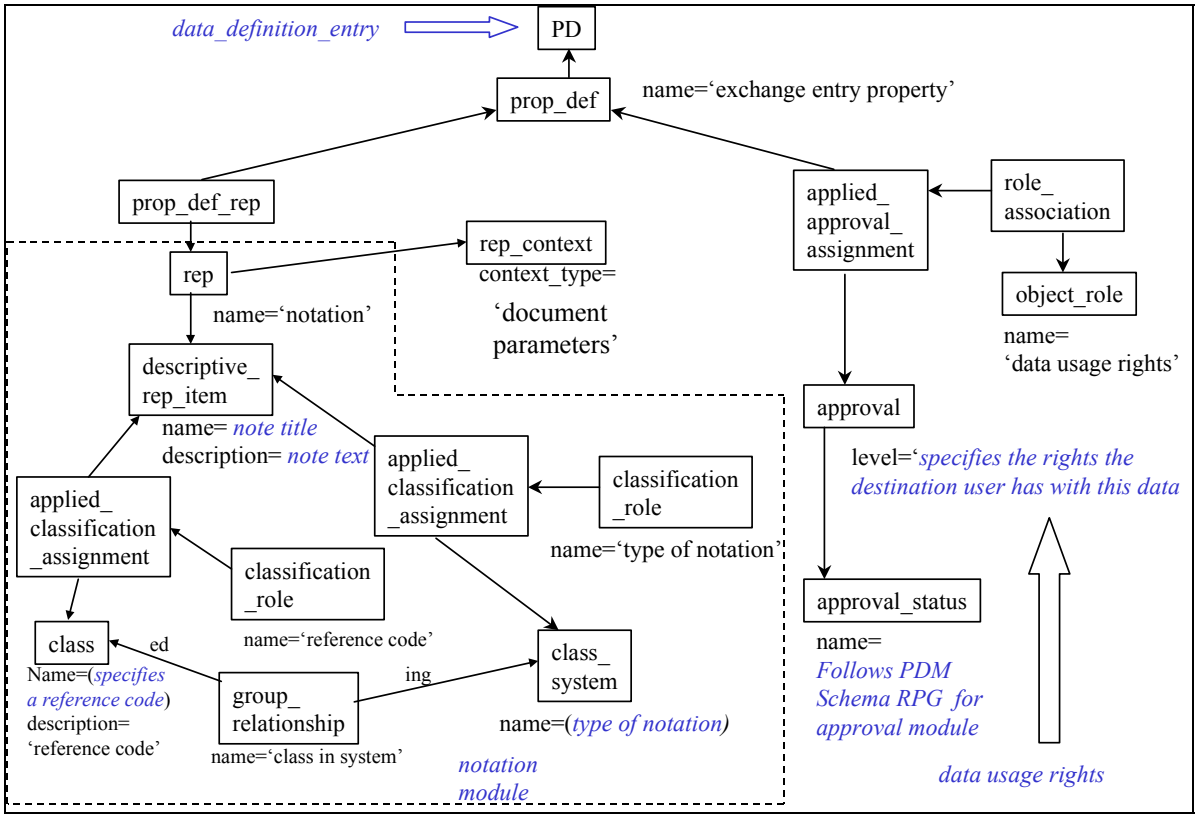


Figure 6.11 - Entry Characteristic (Entry Note, Data Usage Rights)

6.5.1.4 Notation

Notation is a characteristic about an entry, see Figure 6.11. Notation is a human interpretable string of characters. The general concept of notation is defined in its own module. This fourth characteristic uses the notation module. When notation is used for a data_definition_entry, **representation_context.context_type**='document parameters'.

6.5.1.5 Data Usage Rights

Data usage rights are the rights the destination user has with this data. An instance of the entity **approval** is used to capture data rights information, see Figure 6.11. The **approval.level** attribute captures the specific text that describes the data usage rights of the destination user. An example could be to stipulate the data usage rights to be 'Limited'. The corresponding **product_definition** (Data_definition_entry) is associated to this **approval** through a **property_definition** and an **applied_approval_assignment**. The corresponding attributes are constrained to uniquely identify the path. **property_definition.name**='exchange property entry' **object_role.name**='data

usage rights' (associated with **applied_approval_assignment**). The values for **approval_-status.name** will follow the PDM Schema RPG for approval module.

6.5.1.6 Available From – Contract Submission

Available from is an entry characteristic identifying from where the data_definition_entry is accessible, either identified through a contract or some referenced document (see Figure 6.12). The contract submission identifies a contract, the location of the contract submission, and date of submission. The contract is applied to the Data_definition_entry (**PD**) using an **applied_-contract_assignment** with a corresponding **object_role.name** attribute constrained to 'contract submission'. The location of the contract_submision is captured with an **organization** entity and its **organization_address** entity. The organization is applied to the **applied_contract_-assignment** with the corresponding **organization_role.name**='location of contract submission'. The date of submission is captured by applying the date entity to the **applied_contract_-assignment** using an **applied_data_assignment** entity with its corresponding **data_-role.name**='date of submission'. When a time is also required to be captured with the date, apply a **date_and_time** entity to the **applied_contract_assignment** entity using an **applied_date_-and_time_assignment** with a **date_time_role.name**='date and time of submission'.

6.5.1.7 Available From – Document Reference

A reference document also identifies the source where the data_definition_entry is available, see Figure 6.12. This reference document contains the referenced from information. The reference document is mapped to a **product** and associated to the Data_definition_entry (**PD**) through an **applied_document_reference** with the associated **object_role.name** ='source identification'. The identification of the reference document follows the document_identification module RPG.

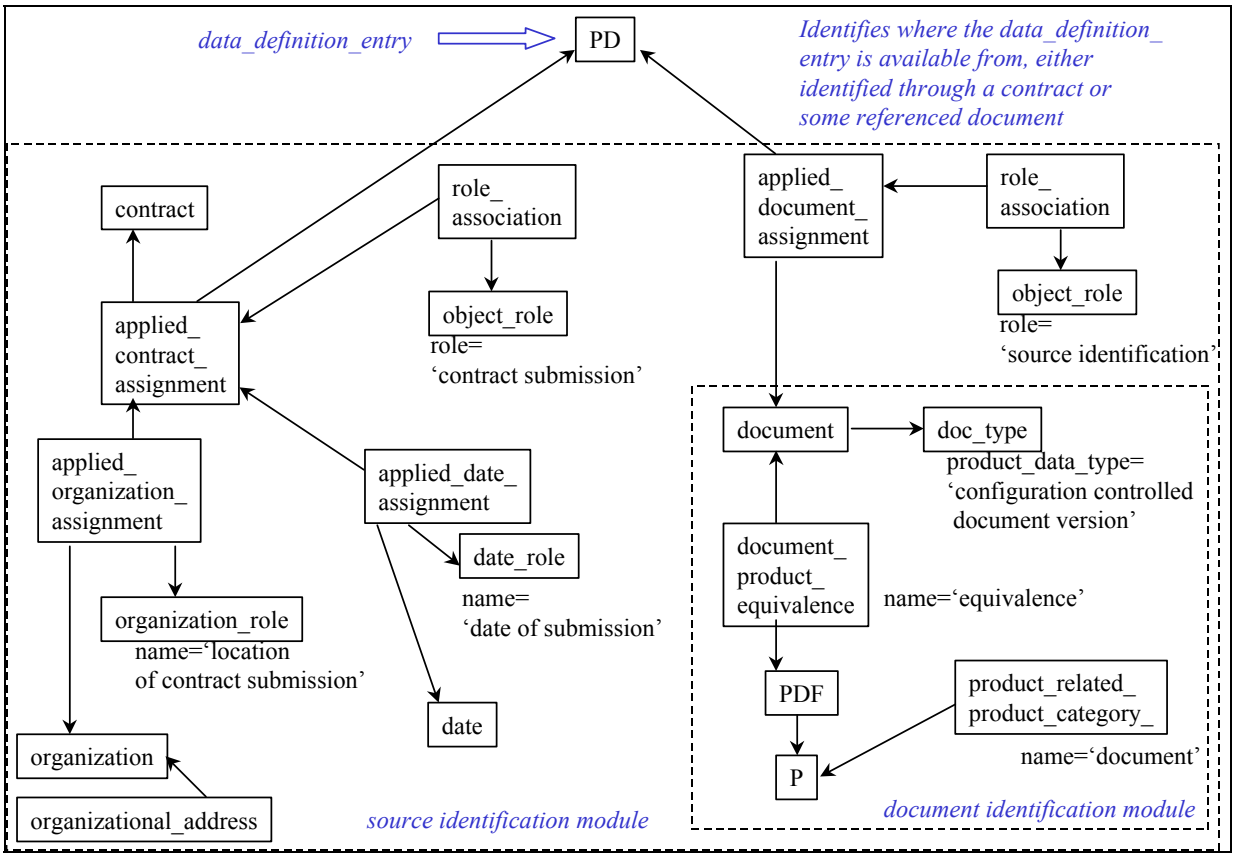


Figure 6.12 - Entry Characteristic (Source Identification)

6.5.1.8 Change Information - Superceded

The superceded characteristic establishes a relationship between a document or part in the current list of entries and the one it replaced from the previous version of an indentured list exchange or document list exchange (see Figure 6.13). This relationship is captured by instancing a path through a **PDR** relationship or a **PDFR** relationship. The entry that is being replaced/superceded is identified with the 'related' attribute in the relationship entity, **PDR** or **PDFR**. The name attributes of the relationship entities are constrained to **PDR.name**='superceded element' and **PDFR.name**='superceded version'.

6.5.1.9 Change Information – Version of the Exchange Management Document an Entry Applies

The version characteristic establishes which version of the exchange management document this entry applies, see Figure 6.13. This will provide the capability to track and record all change history of the exchange process between two enterprises. The version of a particular exchange between two enterprises is not only captured by a PDF but is also an **executed_action** that identifies the change level of a document. The document in this case is one of the types of exchange management documents that list documents to be exchanged. The **executed_action** that is associated to the Data_definition_entry_item (PD) or Data_definition_entry_tdp_element (PD) is the same **executed_action** that was associated to the previous version of the exchange management document. To capture full history of change activity between two enterprises and

executed_action for each previous version of the exchange management document would exist and each would be associated to the appropriate Data_definition_entries. Here a Data_-definition_entry could be identified as being valid in more than one version of the exchange management document. An **applied_action_assignment** with its associated **object_role.name** =’change identification for entry’ is used to connect the Change_identification to the data_-definition_entry_item or data_definition_entry_tdp_element (PD).

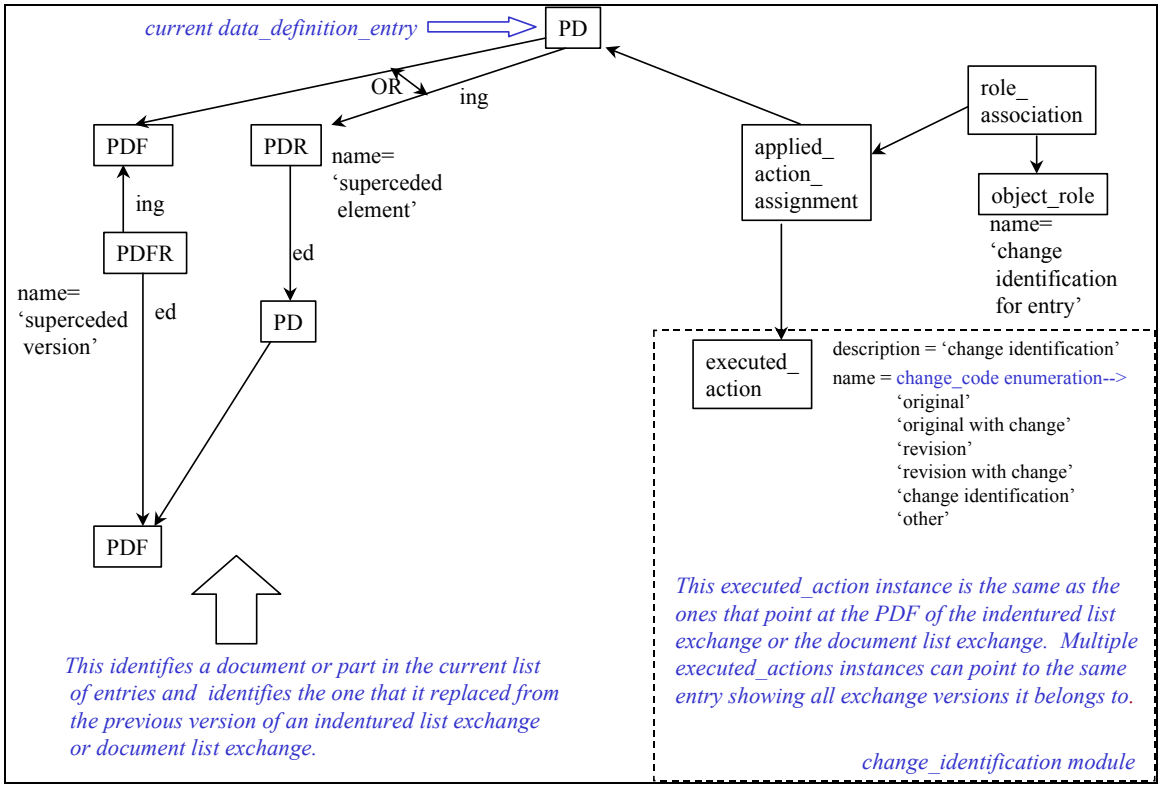


Figure 6.13 - Entry Characteristic (Superceded, Entry Change Level)

The change_identification module is used in this ninth characteristic. The change_identification module section 5.2 will provide guidance for instancing the appropriate entities such as **executed_action**.

NOTE A **PDF** with its associated **executed_action** having its attribute **name**=’change identification’, signifies that this is only change information for this **PDF**(version) of the document. This could be use to facilitate a level of data change within an exchange scenario.

6.5.1.10 Associated Files

The associated files characteristic identifies the set of files associated with a data definition entry (**PDWAD**) see figure 6.14. These files represent the document or part identified by the data definition entry. The configuration of these files is managed based on their respective document or part.

NOTE A document version may have multiple representation, (e.g., Raster and Postscript). Each of these document version representations will be instantiated by a unique **PD** or **PDWAD**. When there is a file or set of files that make up a particular representation, they are associated with one **PDWAD**.

A document_file is instantiated to identify a file. The file identification module will provide guidance in instantiating file information.

6.5.1.11 Document Format and Size

The document format and size characteristic identifies two basic document properties that are applied at the Data_definition_entry level, see Figure 6.14. The two document properties are document format and document size. These document properties are in the document_properties module and the file_properties module. To get appropriate byte size for the total document representation the set of **measure_rep_items** the **representation** points to must be summed, (this is also true for page count). This set of **measure_rep_items** could be a set of one. If there is a set of **measure_rep_items**, each **measure_rep_item** represents the value based on a file.

NOTE multiple files could be needed to represent one complete instance of a document.

A file is a collection of data in a particular format. A file_identification module has been defined within the PDM Schema module set. The exchange file identification is the same as the file_-identification module with additional types of parameters. Particular instances of document format and document size can be associated to both a file and document representation in the same exchange.

EXAMPLE part 21 for capturing byte size and page count :

```
#1000023=MEASURE_REPRESENTATION_ITEM('file size',COUNT_MEASURE(3.0),#1000050);
#1000024=MEASURE_REPRESENTATION_ITEM('page count',COUNT_MEASURE(3.0),#1000051);
#1000050=CONTEXT_DEPENDENT_UNIT(#1000060,'BYTE');
#1000051=CONTEXT_DEPENDENT_UNIT(#1000060,'COUNT');
#1000060=DIMENSIONAL_EXPONENTS(0.0,0.0,0.0,0.0,0.0,0.0,0.0);
```

6.5.1.12 Delivery Accounting Reference Documents

The delivery accounting reference characteristic identifies delivery accounting reference documents that are related to the data definition entry **(PD)**, see Figure 6.15. Examples of delivery accounting reference documents are letters of transmittal and material receiving inspection forms. The delivery accounting reference document is identified by a **produc**t using the part_identification module. The delivery accounting reference document is associated to the data definition entry (**PD**) through a **PDR** with its **name** attribute constrained to 'delivery accounting reference'.

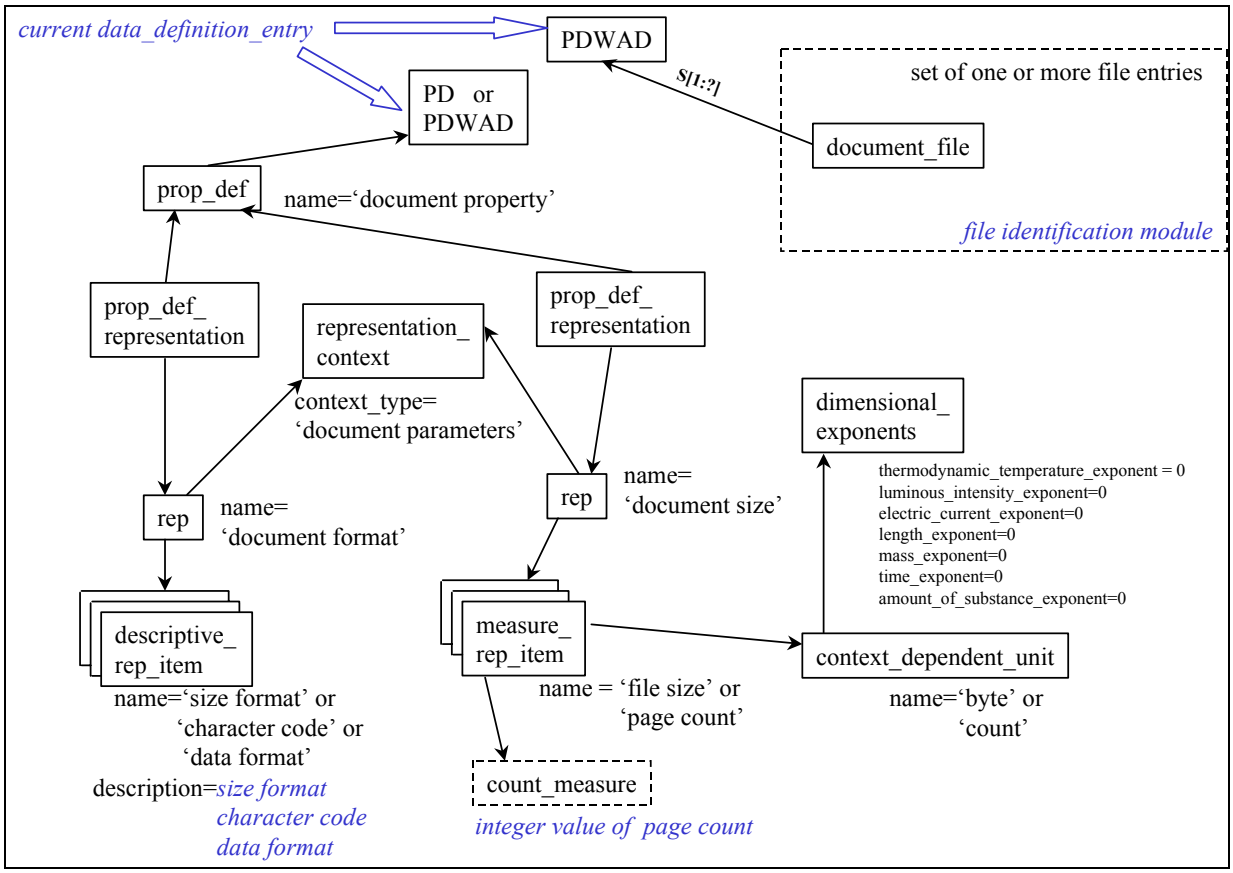


Figure 6.14 - Entry Characteristic (Document Format, Document Size)

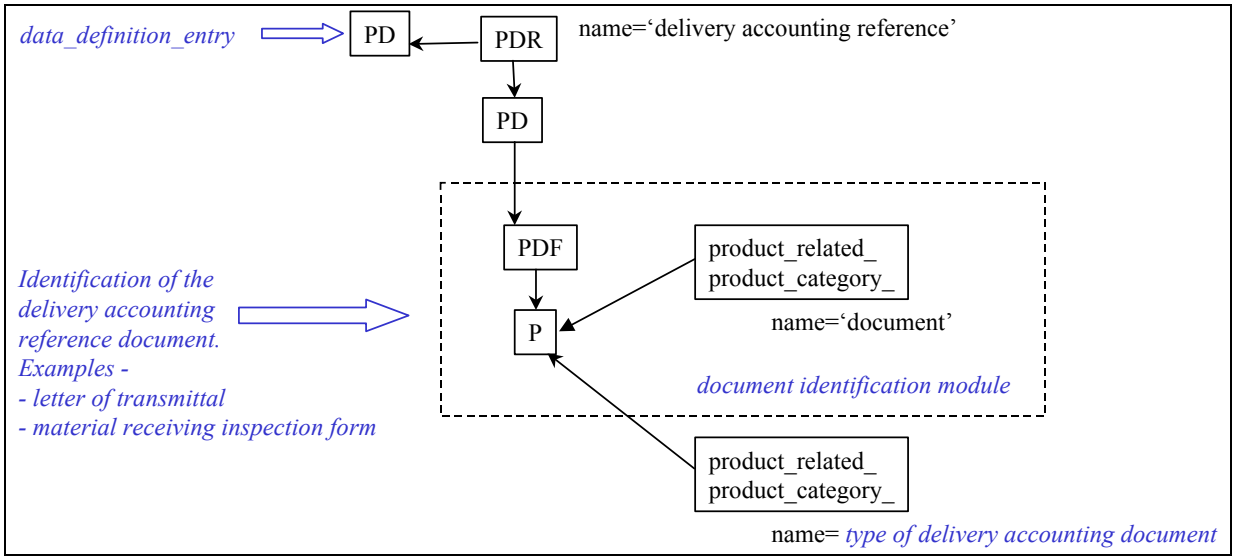


Figure 6.15 - Entry Characteristic (Delivery Accounting Reference)

6.6 Exchange File

An Exchange File is a file (copy of a file) that is being transferred to another location. These additional file parameters include:

— Additional File Size Characteristics;

— Additional File Format Parameters;

— File Security Classifications;

— File Distribution Notices;

— File Change Status;

— Access, Source System and Destination System Parameters and

— Include Flag.

Access file parameters of path information and storage node identification will also be described in this section.

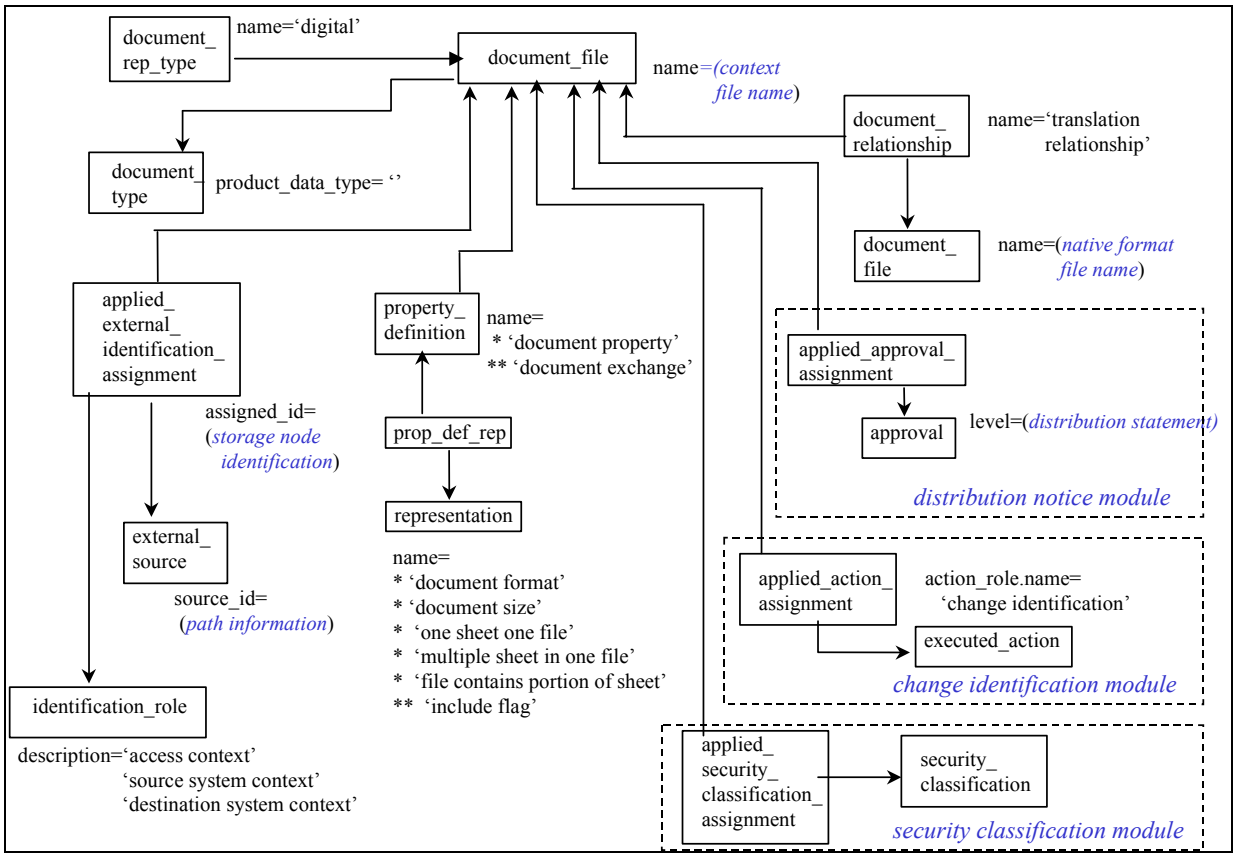


Figure 6.16 - Exchange File Parameters

A file is mapped to the integrated resources as a subtype of both a **document** and **characterized_object**. This new entity is called a **document_file**. Figure 6.16 shows the high-level results of mapping these requirements to the integrated resources. Each of the instantiations of these file parameters is shown in more detail in figures Figure 6.17 through 25. In Figure 6.16, there are some string constraints that are pared up by the use of a '*' or '**' notations.

6.6.1 Additional File Size Characteristics

Figure 6.17 describes the file size parameters. The first two parameters, byte size and page count, can be applied to both a file and a document representation (one or more files that make up a specific representation of a document version). In Figure 6.17 the string constraints in **representation.name**, **representation_context.context_type**, and **measure_representation_-item.name** are matched up by using the notation '*', '**', '***', or '****'. The **measure_-representation_item.name** and **context_dependent_unit.name** string constraints are matched up with block arrows.

The file size parameters that are captured are file size, page count, internal division count, internal division type, sheet portion number, and number of subdivisions of the sheet. The file size is the byte size of the computer disk space required for a particular file. The page count specifies the number of physical pages or sheets that are contained in a particular file.

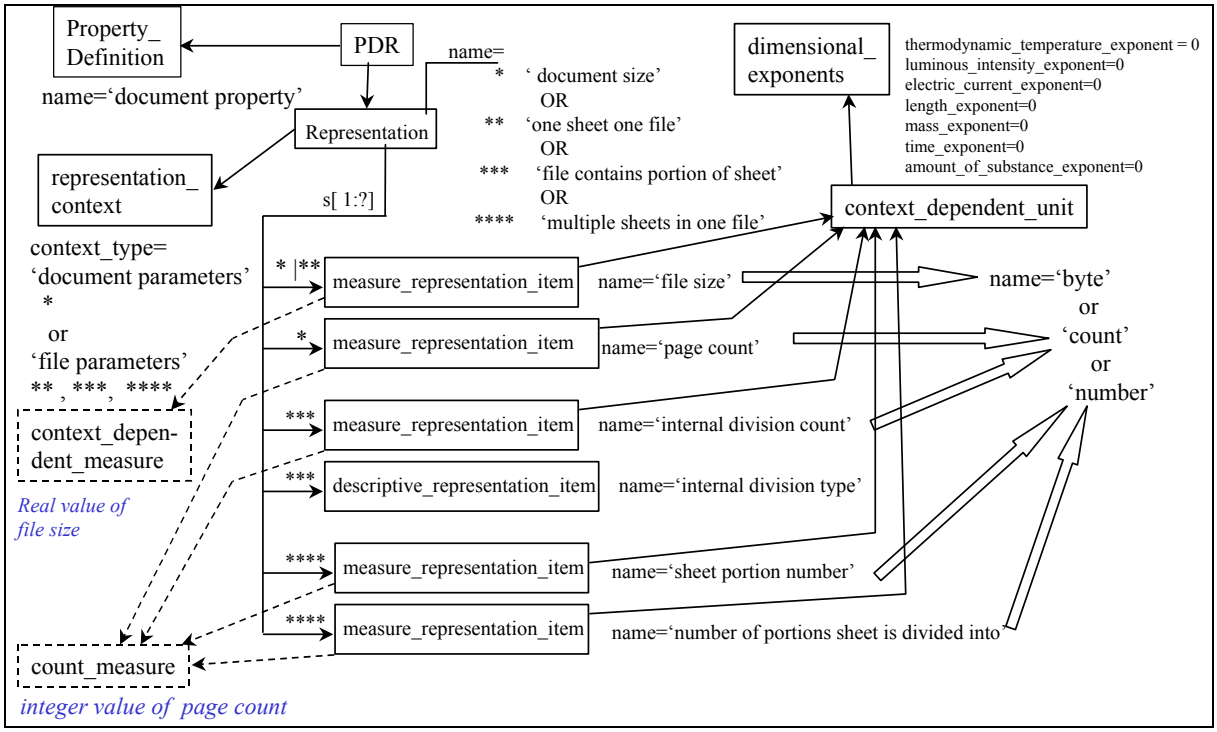


Figure 6.17 - File Size Parameters

The next set of file size characteristics support the needed parameters for when a file contains many sheet representations or only a portion of a sheet representation. An example of this could be described by capturing a ten-sheet drawing in a collection of raster files. First, sheets one through four are small sheets and are scanned together and captured in one raster file. This raster

file would have the **measure_rep_item.name**='internal division count' and the **count_measure** value of 4. This raster file could also have a **descriptive_rep_item.name**='internal division type' and **descriptive_rep_item.description**='sheet'.

NOTE The **descriptive_rep_item.description** string will vary such as 'page' for books, and 'view' or 'layers' for geometry models.

Sheet five of the example drawing is a very large sheet, and therefore is scanned in three sections, each requiring a raster file. Each of these three raster files would be associated with a **measure_rep_item.name**='number of portions sheet is divided into' and a **count_measure** value of 3. Then each of these three raster files would be associated with its own particular **measure_-rep_item.name**='sheet portion number' and **measure_count** value of 1, 2, and 3, respectively. These **measure_count** values specify the ordinal number for assembling the files for the fifth sheet of the example drawing.

6.6.2 Additional File Format Parameters

Format parameters shown in Figure 6.14 can also be applied to a file. Instead of the **property_-definition** entity pointing to **a product_definition**, the **property_definition** points to a **document_file**. Figure 6.19 shows example values of these format parameters identified in Figure 6.14.

The additional format parameters that can be associated with a file are the release date of the format and the identification of the format's change or revision level. These parameters should be used as needed to uniquely identify the file format. These parameters are contained in the change identification module, section 5.2. Figure 6.19 shows how the change identification module is associated with a **representation** construct, which is the focal point for capturing file format information.

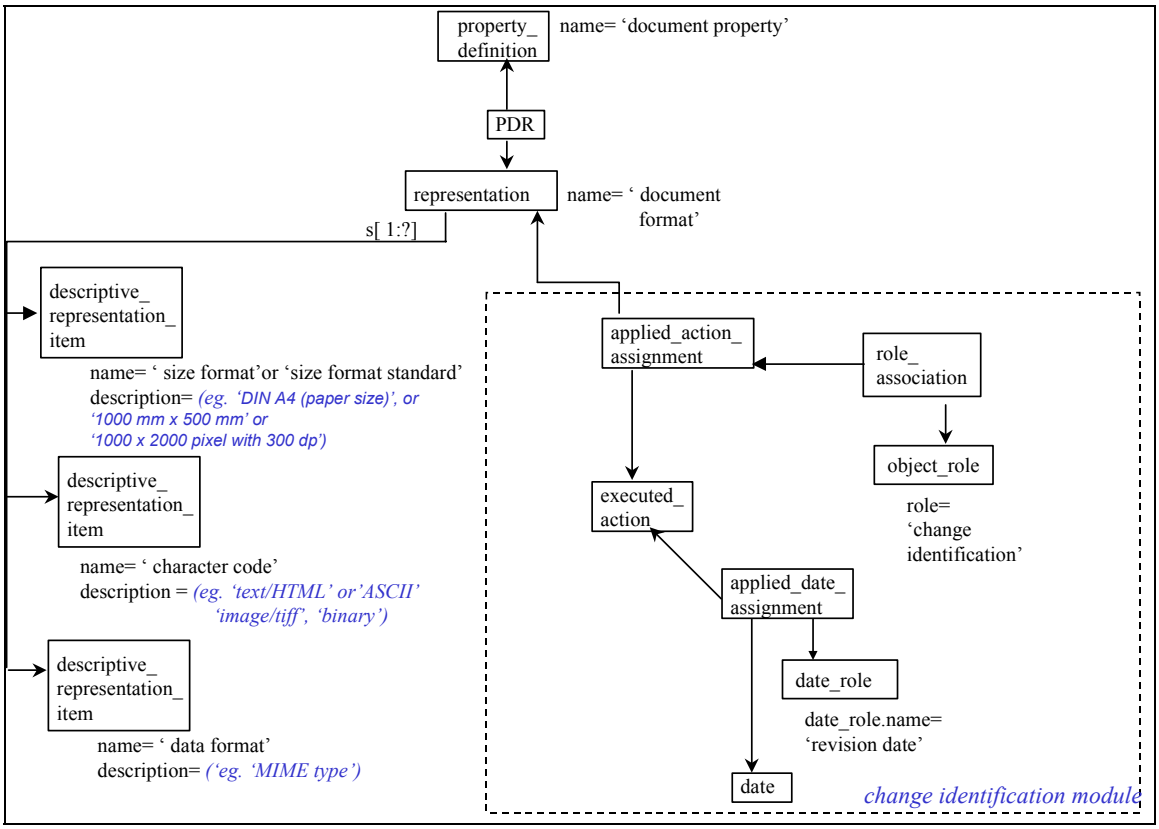


Figure 6.19 - File Format Parameters

6.6.3 File Security Classification

A file may have a security classification placed on its content/title. Figure 6.19 shows the constructs needed to capture this information. To place a security classification on a file's content, associate the **security_classification.purpose**='item security classification' with the associated **security_classification_level.name** containing the value of the classification. To place a security classification on a file's title associate the **security_classification.purpose**='title security classification' with the associated **security_classification_level.name** containing the value of the classification.
The date of when the security classification became effective and was removed is identified by the **date_role.name**='classification date' or 'declassification date', respectively. Both the security classification module and the data time module is needed to satisfy file security classification requirement.

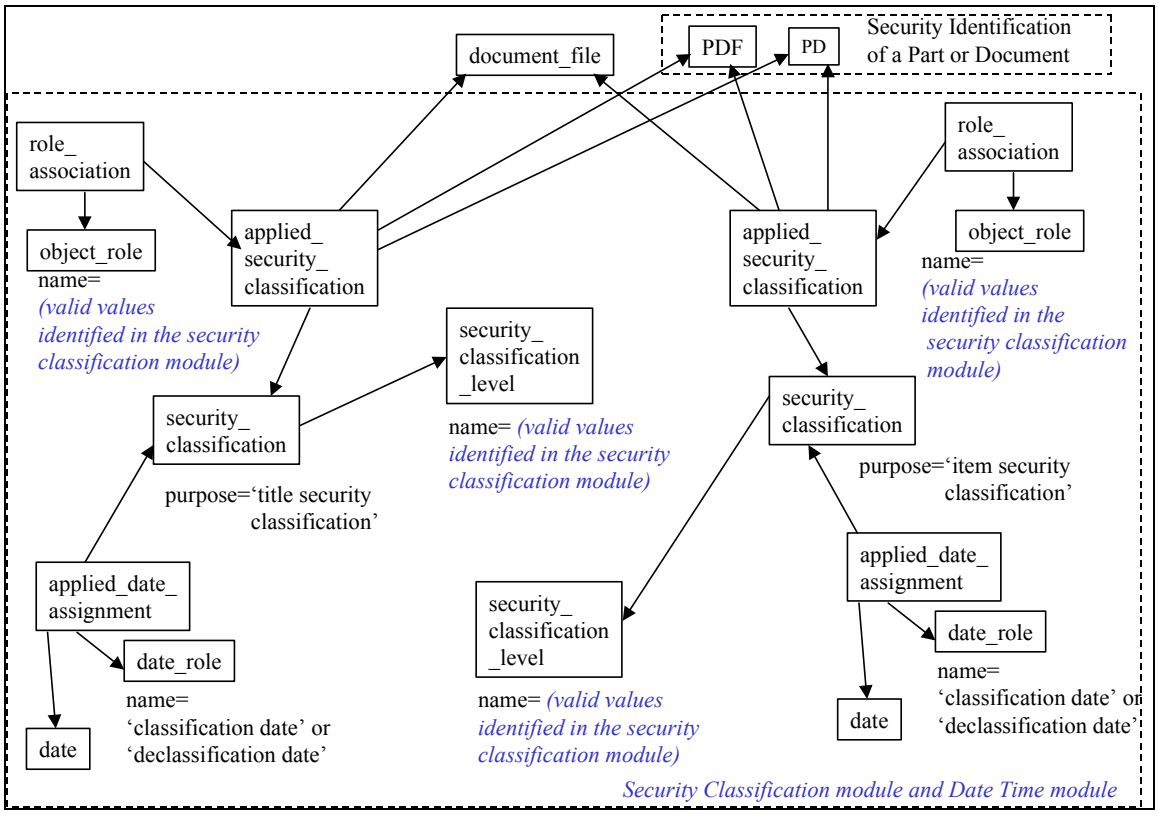


Figure 6.19 - Security Classification (for File, Part, Document)

6.6.4 File Distribution Notice

The definition of a distribution notice is found in the distribution notice module. For a file, a distribution notice is the allowable industrial or organizational circulation of information for the organization receiving the file. The distribution notice information is associated to a file through an **applied_approval_assignment** construct with an **object_role.name**='distribution notice', see Figure 6.20.

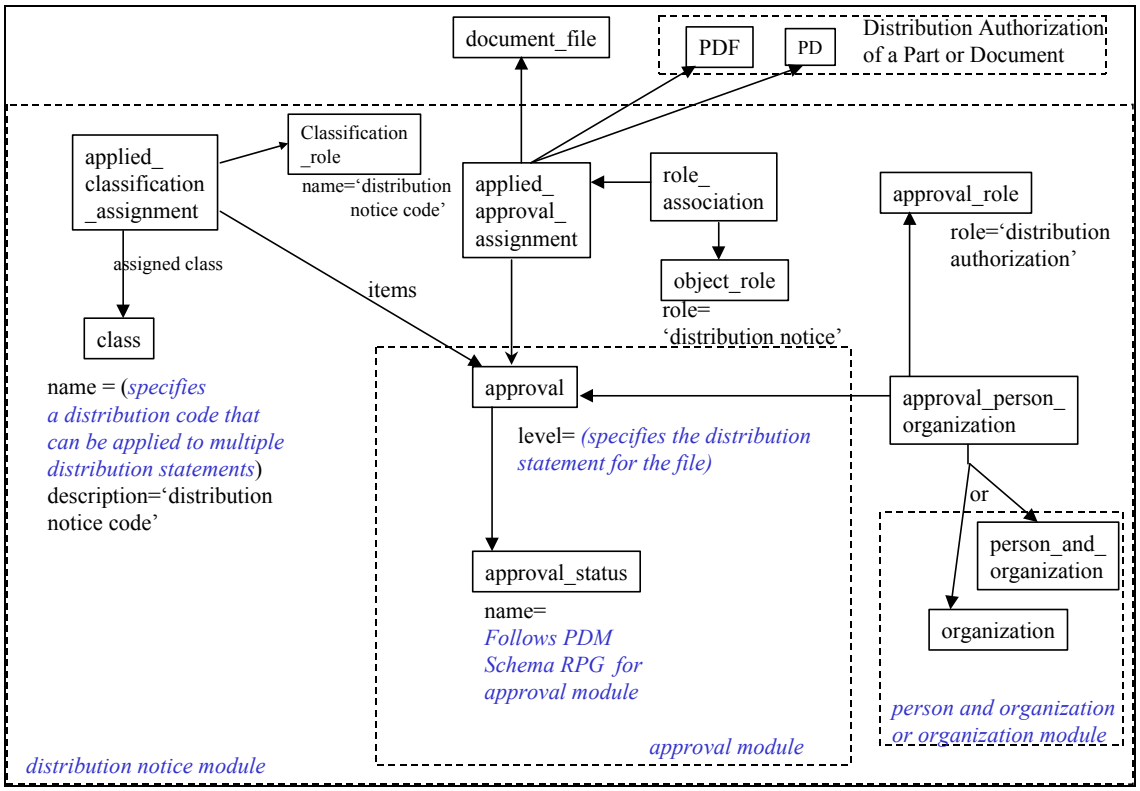


Figure 6.20 - Distribution Notice (for File, Part, Document)

6.6.5 File Change Status

File change status specifies two concepts, the version of the file and the change identification. The first, file version, is captured through the use of the alias identification module, in PDM schema V1.1. The second, change identification, is captured through the change identification module, section 5.2.

Figure 6.21 shows how these change status concepts are associated to a file. The identification of a file is made through the instance of a **document_file** construct. A version is placed on a **document_file** by an associated **assigned_identification_assignment.assigned_id**; the associated **identification_role.name**='version'.

NOTE for each version of a file a new instance of **document_file** must be created.

The change identification parameters for a file are placed on a **document_file** by an **applied_action_assignment** with an associated **object_role.name** ='change identification'. These change identification parameters are the same parameters that are used for a document.

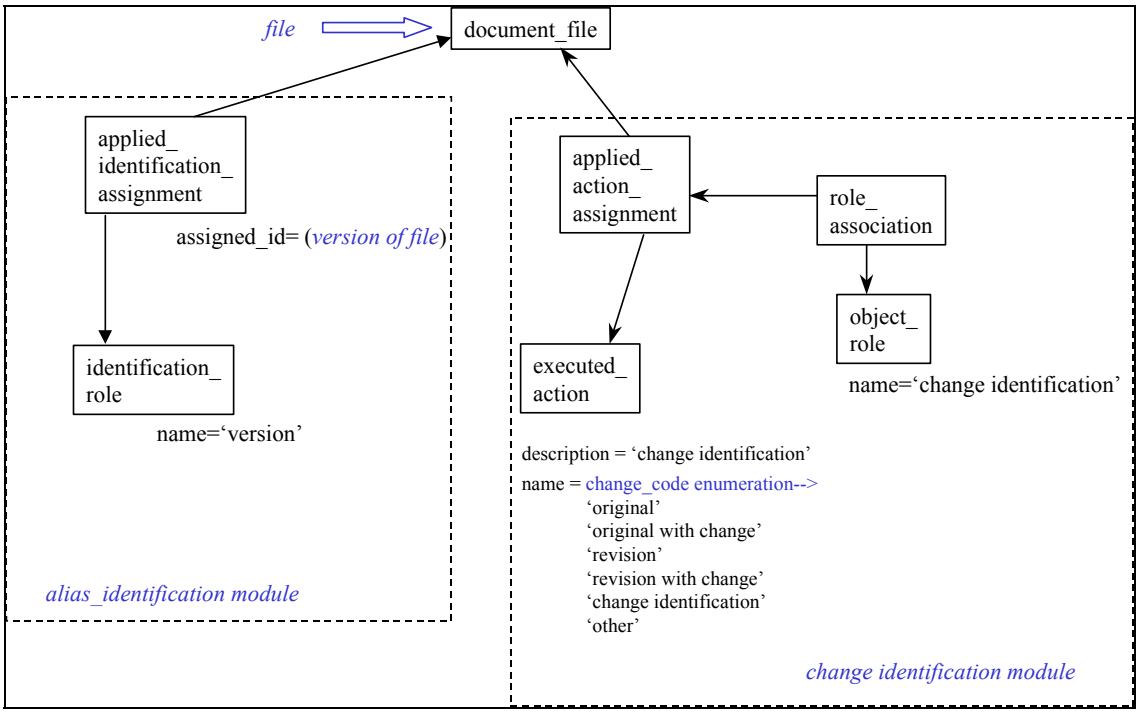


Figure 6.21 - File Change Status

6.6.6 File access, source_system and destination_system

The identification of how to access a file, identify the source system of a file and identify the destination system of a file are captured using the same set of construct types. These files can be digital or can represent physical things, like the hardcopy of a document or a physical model of a part, as described in the PDM schema Users Guide V1.1 (or higher). Figure 6.22 shows how access, source_system and destination_system parameters are instantiated for a digital file and Figure 6.23 shows how they are instantiated for a physical thing.

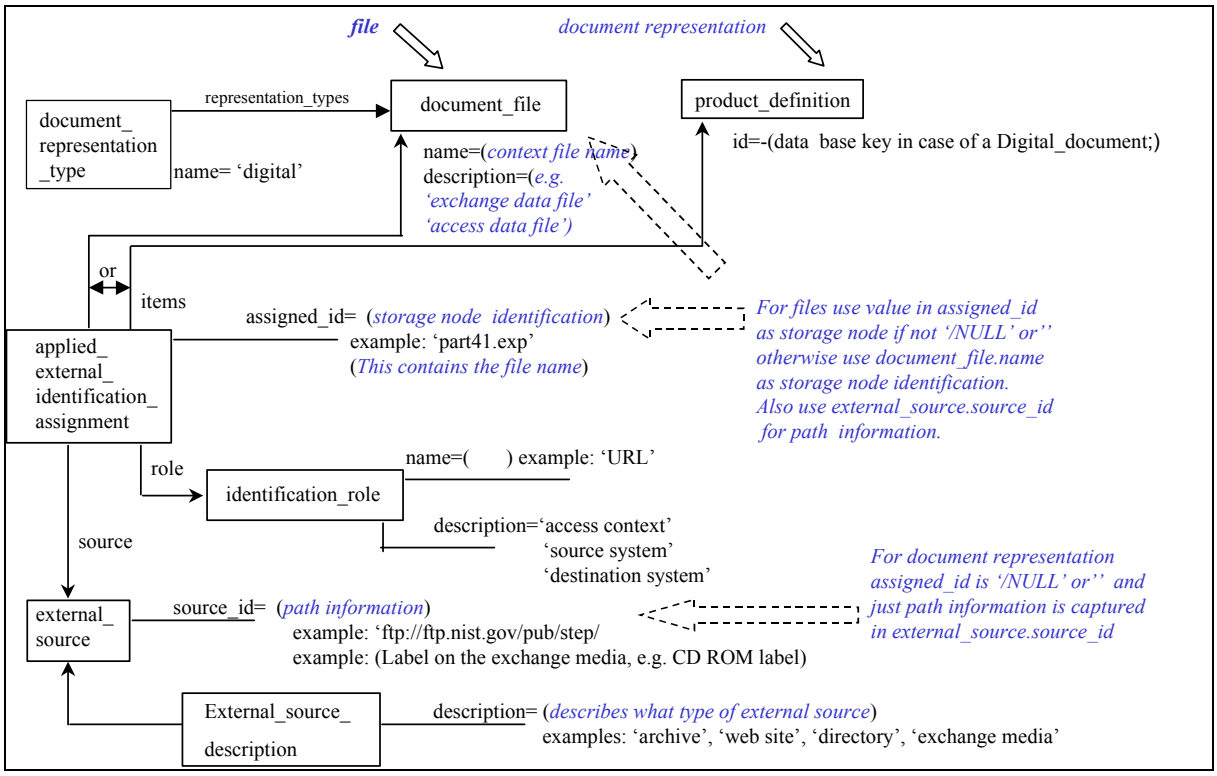


Figure 6.22 - Digital File Access, Source, Destination Parameters

The parameters needed to access a file include storage node identification and path information. Typically this information is captured using a Universal Reference Locator (URL) notation. The storage node identification is captured in an **applied_external_identification_-assignment.assigned_id** attribute. The associated **identification_role.description**='access context' depicts that this **applied_external_identification_assignment** is identifying access information. The **identification_role.name** captures the name or type of how this information is being captured. The example of identification_role.name in the Figure 6.22 is 'URL'. The path that is needed to identify the system the storage node is located in is captured in the **external_-source.source_id** attribute. The associated **external_source_description.description** describes what type of external source is being used.

Capturing the source and destination system of a file is similar to accessing a file, except that the **identification_role.description**='source system' or 'destination system'.

The common parameters are the following:

— The identification of the computer system that the file came from (source) or is going to (destination) or is available to down load a copy from (access).

— The name of the file on the source computer system or what the destination computer system should name the file or what the name of the file is on the system that can be accessed to down load the file.

These source and destination parameters provide a way to synchronize the naming of a file when the name is different at its source, being exchanged, and when it is copied into the destination system. The parameters allow for the destination system of a particular file to be clarified for a receiving application.

In many cases the file name being exchanged (captured in **document_file.name**) is the same as the storage node identification. When this is the case it is allowable to place a '/NULL' or a blank string '' in the **applied_external_identification_assignment.applied_id** attribute. This will signify that the **document_file.name** can be storage as the storage node identification.

When just path information needs to be captured for a document occurrence or a shape model occurrence, the **applied_external_identification_assignment** can be associated directly to a product_definition. In this case the **applied_external_identification_assignment.applied_id** is either '/NULL/ or a blank string ''.

If the case occurs when a file needs to be related to another file already in the destination system, an **identification_assignment_relationship** can be instantiated with an addition **applied_-external_identification_assignment** and **document_file** denoting the file in the destination system. An example of this would be when a file containing a drawing border is already on the destination system and a file containing the part views and title information of a drawing are being sent. This additional **applied_external_identification_assignment** and **document_file** for the drawing border file could help synchronize these files for this drawing.

There are times when the exchange of hardcopies of documents or location of hardcopy documents or physical models need to be captured and managed. The same constructs are used as if it where a digital file. Figure 6.23 shows the how to capture location information for physical hardcopies or models.

EXAMPLE - An scenario where this may apply could be:

A paper drawing is being sent to a part supplier along with a digital 3D model of the part. From an exchange management perspective the exchange of both the paper drawing and digital 3D model need to be tracked. The STEP exchange management file that is sent to the supplier would identify the drawing and the 3D model, plus the knowledge that the drawing is non-digital. (Non-digital is captured by instantiating **document_representation_type.name='physical'.**) If the paper drawing was shipped via Federal Express package delivery service, this knowledge could be captured.

EXAMPLE -

- The delivery companies name would be captured as **external_source.source_id**='Federal Express'.
- The description of the external source would be captured as **external_source_description.-description**='package delivery'.
- The tracking number for the package would be captured in the **applied_external_identification_-assignment.assigned_id**= 'fedex 2324049' with the **associated identification_role.role**='tracking number'.

Other examples of capturing locations of physical things (documents and models) are the following:

Location of a book in a library;
Location of a specific volume of an encyclopedia;
Location of a physical model in a warehouse.

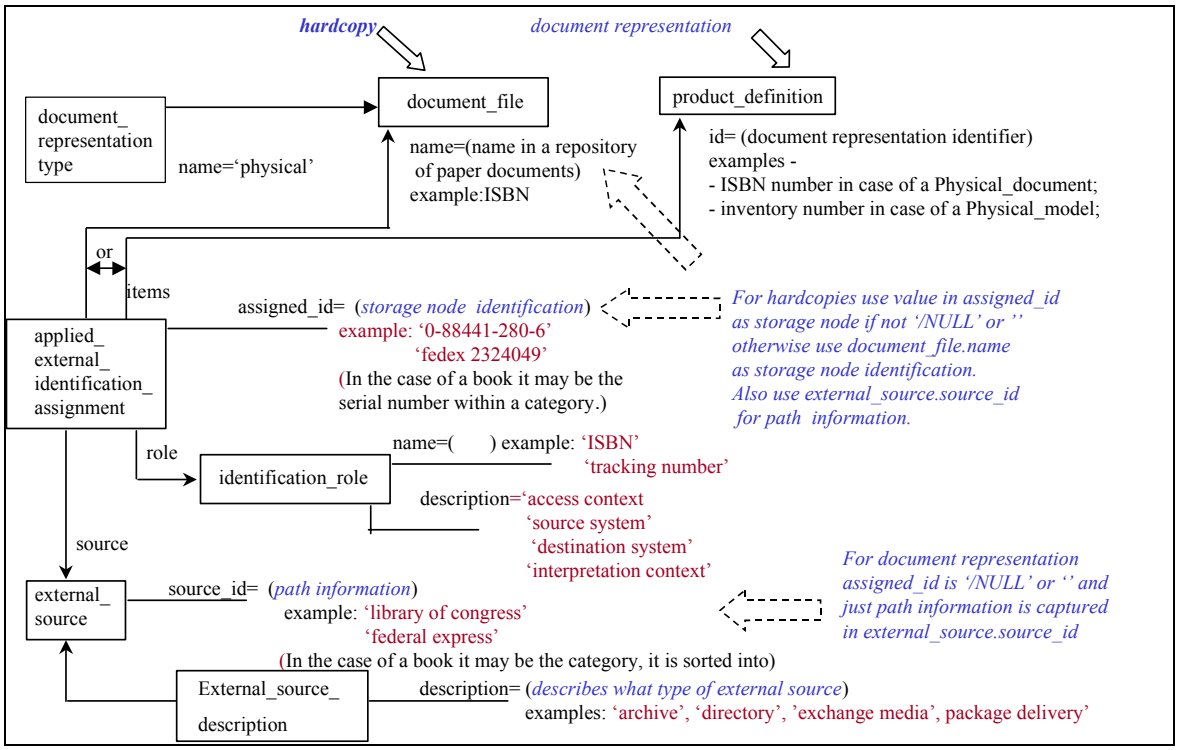


Figure 6.23 - Physical Document and Physical Model Location Parameters

6.6.7 Include Flag

The include flag for a file specifies whether or not the file is included or furnished as a portion of this exchange. The include flag for a file is similar to the include flag for data definition entry, section 4.5
The only difference is that the property_definition is associated with a document_file instead of a product_definition as shown in Figure 6.24.

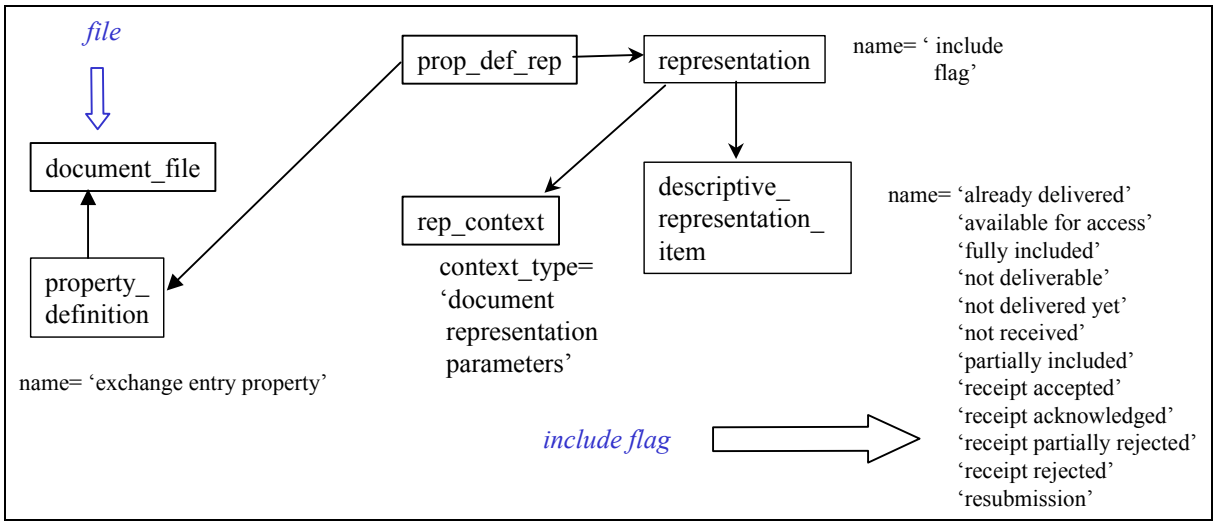


Figure 6.24 - File Include Flag

6.7 Exchange Management Header

The exchange management header is the configuration, data management, and usage information necessary to manage and control the collection of data being identified with in an Exchange Management document. The Exchange Management Header contains information such as the business purpose of the exchange, references other business documents, exchange date of transfer, and exchange destination.

The exchange management header module is existence dependent on the document header module, section 4.1. While the document header module is generic for any document, the exchange management header module supports the needed information for the three types of exchange management documents, (Exchange Management Using Product Structure, Exchange Management Using Document List, Exchange Management Using File List).

The parameters that make up the exchange management header (excluding the generic document header parameters) are the following:

— Exchange Management Document Identification;

— Date of Transfer;

— Delivery Accounting References;

— Destinations;

— Procurement References;

— Reason for Exchange.

6.7.1 Exchange Management Document Identification

The exchange management document identification specifies that this particular document type contains exchange management information. Figure 6.25 shows that a **product_related_-product_category**, with its name attribute equal to 'data definition exchange,' specifies its related product construct and is an exchange management document.

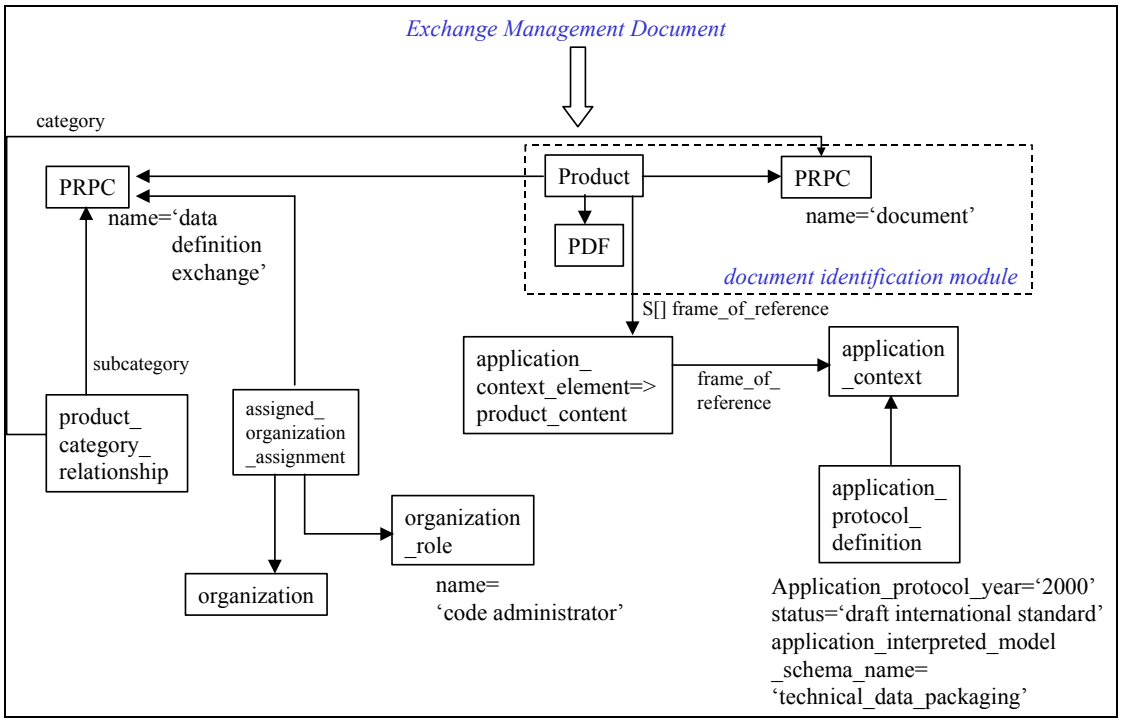


Figure 6.25 - Identification of Exchange Management Document

6.7.2 Date of Transfer

The date of transfer specifies the date and time when the responsible activity initiated the transfer of the exchange management document and associated documents and files. The date of transfer need not be specified for a particular exchange management header. Figure 6.26 shows where a date of transfer is applied and instantiated.

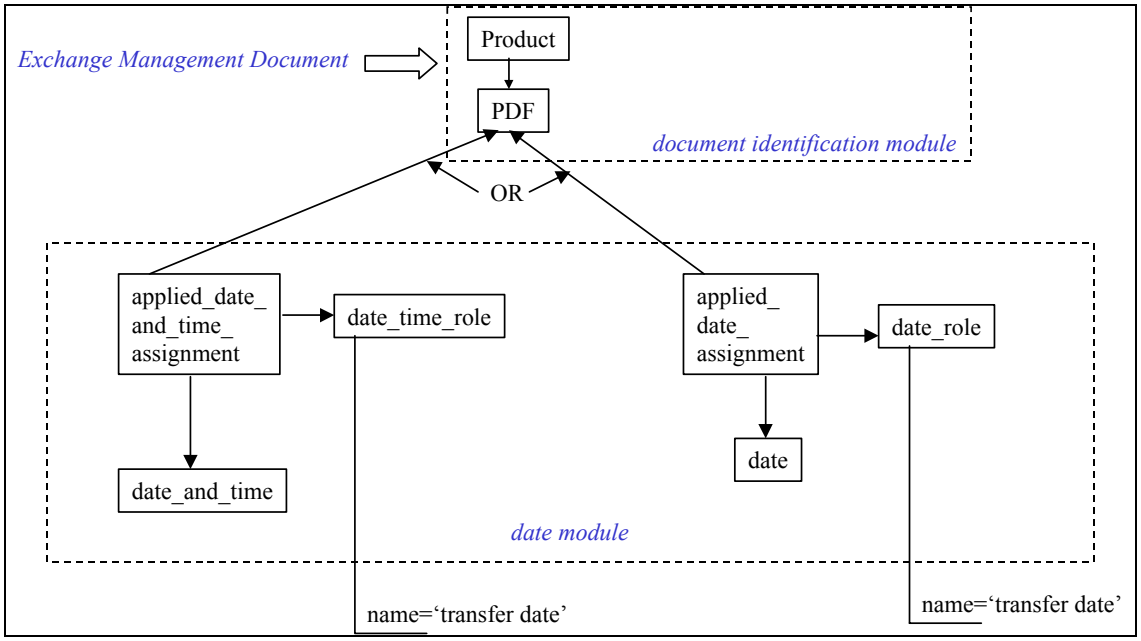


Figure 6.26 - Date of Transfer

6.7.3 Delivery Accounting References

The delivery accounting references specify the information required for the business rationale for the delivery of the exchange management information. The delivery accounting references need not be specified for a particular exchange management header. There may be more than one delivery accounting references for an exchange management header. (EXAMPLE - Delivery accounting information could be letters of transmittal, Purchase Orders, and Material Inspection Report Forms.) Figure 6.27 shows how delivery accounting references are instantiated.

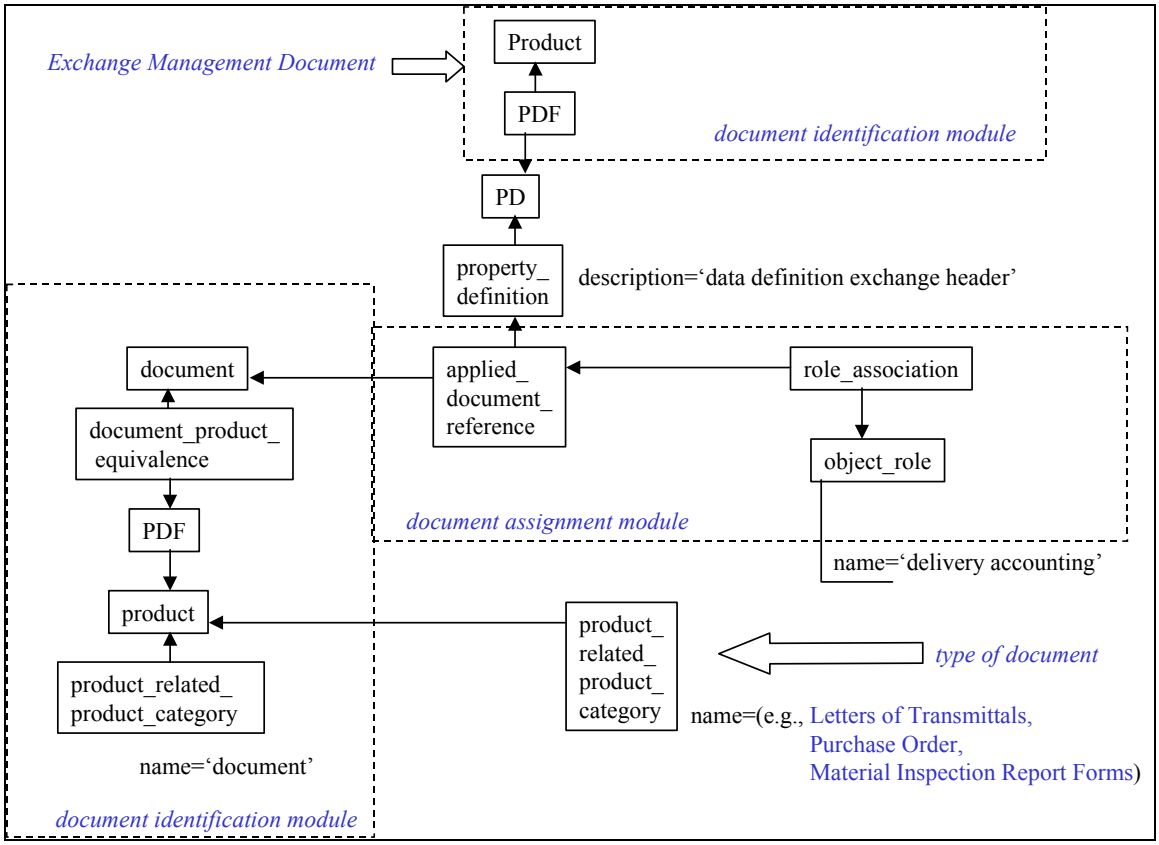


Figure 6.27 - Delivery Accounting Reference

6.7.4 Destinations

The destinations specify the target organizations where the exchange management document and associated documents and files are transmitted. The destinations need not be specified for a particular exchange management header. There may be more than one destination for an exchange management header. Figure 6.28 shows how destinations are instantiated as an organization destination.

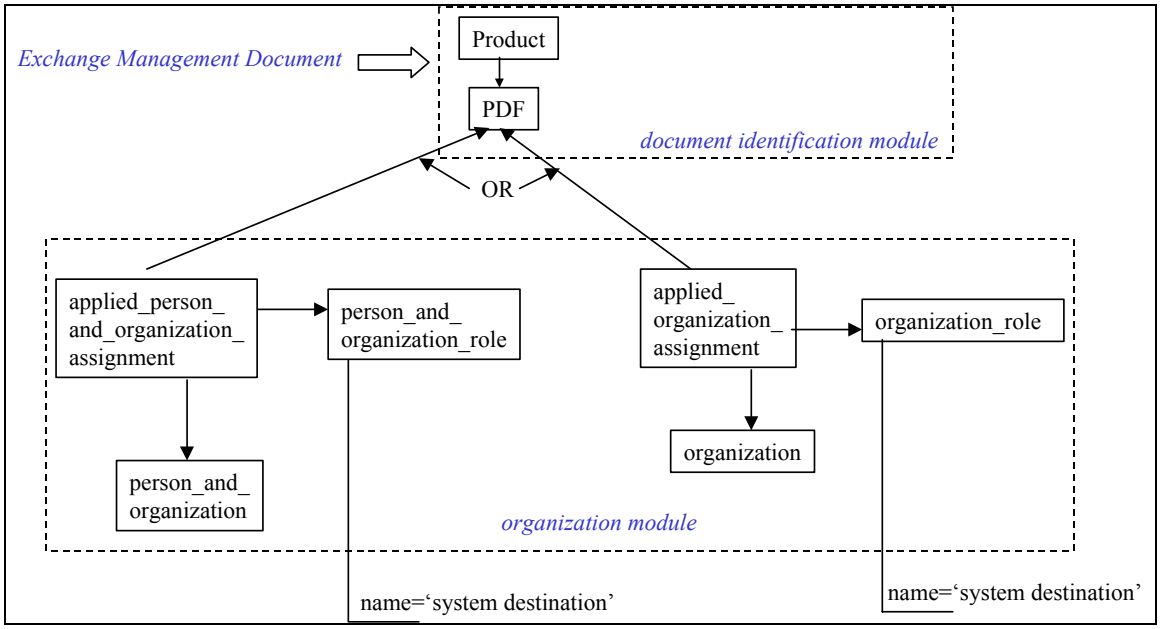


Figure 6.28 - Destinations

6.7.5 Procurement References

The procurement references specify a part identification or a document identification that was used as justification for development of the exchange management document. The justification is a business purpose that identifies a document or part that was provided by the organization procuring the data contained in the exchange management document. The business purpose permits the association of different exchange management files in a hierarchical or other business-purpose, defined method. The procurement_references need not be specified for a particular exchange management header. There may be more than one procurement_references for an exchange management header. Figure 6.29 shows how procurement references are instantiated.

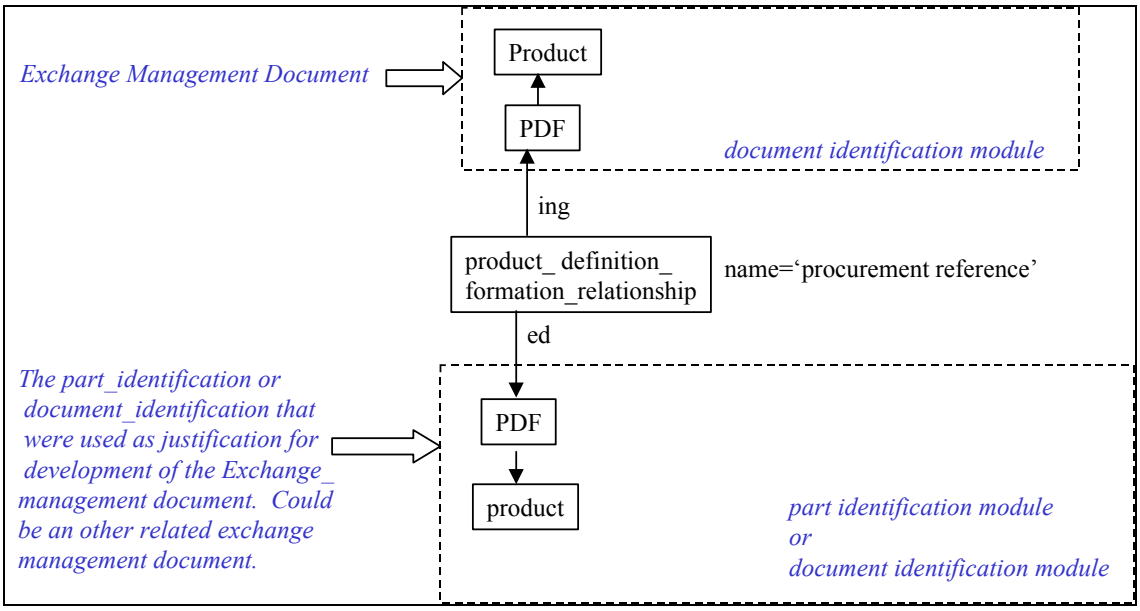


Figure 6.29 - Procurement References

6.7.6 Reason for Exchange

The exchange reason specifies the purpose for the exchange management information exchange. The exchange reason is made up of 3 parameters that can be instantiated independent from one another. The three parameters are based reason for the exchange, purpose of exchange, and the item being exchanged. Figure 6.30 shows how these three exchange reason parameters are instantiated.

The base reason provides a set of general exchange reasons that exchange partners may use to trigger other processes based on the reason's meaning. The following are the set of base exchange reasons.

6.7.6.1 acknowledge receipt of delivery

The exchange that is a correspondence from the receiver to the original sender acknowledging the receipt of the data provided by the sender.

6.7.6.2 engineering design analysis

The exchange that is a general purpose exchange of design information for general analysis and technical interchange. The exchange does not fulfill contractual obligations and is considered not formal.

EXAMPLE An exchange for concurrent engineering design analysis or comparisons is not a formal contractual obligation

6.7.6.3 final design review

The exchange that is a formal exchange and is intended to fulfill contractual or other obligations. The receiver of the package is expected to conduct a formal review of the package and assess the final design, methods, and standards used in design. The receiver then provides an acceptance or rejection of the design in terms of technical engineering content (not acceptance or rejection of the design documentation as a deliverable).

6.7.6.4 initial data submittal

The exchange that is submitted in partial fulfillment of contractual obligations requiring the delivery of design data. The initial submittal is the first formal delivery of configuration managed design data which comprises the design "baseline". The receiver of the package is expected to conduct a formal review of the package and assess the final design, methods, and standards used in design to ensure it meets specified requirements (e.g., the support of redesign, repair, remanufacture, or reprocurement). The receiver then provides a formal disposition of acceptance or rejection of the design in terms of both the technical engineering content and design documentation.

6.7.6.5 interim engineering design review

The exchange that is a formal exchange intended to fulfill contractual or other obligations. The receiver of the package is expected to assess the preliminary design concepts, methods, and standards and provide general feedback to the sender.

6.7.6.6 procurement design package

The exchange that contains the formally released or managed technical design data required to perform a part procurement activity. The package includes a complete design package from which the receiver can manufacture or provide an item.

6.7.6.7 provisioning data submittal

The exchange that is submitted in partial fulfillment of contractual obligations requiring the sender to provide design data (at negotiated levels of design documentation completeness) which enables provisioning analysis on the part.

6.7.6.8 request for proposal

The exchange that is provided to support the development of a requested proposal involving the designed item.

6.7.6.9 request for quote

The exchange that is provided to support the development of an estimate involving the designed item.

6.7.6.10 revision update data submittal

The exchange that is submitted in partial fulfillment of contractual obligations requiring the delivery of design data. The revision update data submittal is a formal delivery of configuration-managed design data (subsequent to the initial data submittal) which revises the design

"baseline". The receiver of the package is expected to conduct a formal review of the package and assess the final design, methods, and standards used in the design to ensure it meets specified requirements (e.g., the support of redesign, repair, remanufacture, or reprocurement). The receiver then provides a formal disposition of acceptance or rejection of the design in terms of both the technical engineering content and design documentation.

6.7.6.11 specification design package

The exchange that contains the formally released-managed technical design data required to perform a design procurement activity. The package includes a complete set of design specifications (e.g., specification control drawings, associated lists, material and process specifications, etc.) from which the receiver can design the desired item.

The purpose of exchange parameter allows business partners to agree upon a set of valid strings to be communicated for auto-processing purposes or for visual interpretation. This string may augment the base reason for the exchange or it may stand alone.

The item the exchange is applicable to is a reference to the information being exchanged that pertains to a part. There can be more than one part related to the exchange in this manner. Assigning a part or parts to an exchange this way was intended to provide the relative parts for an exchange management using document list method and an exchange management using file list method.

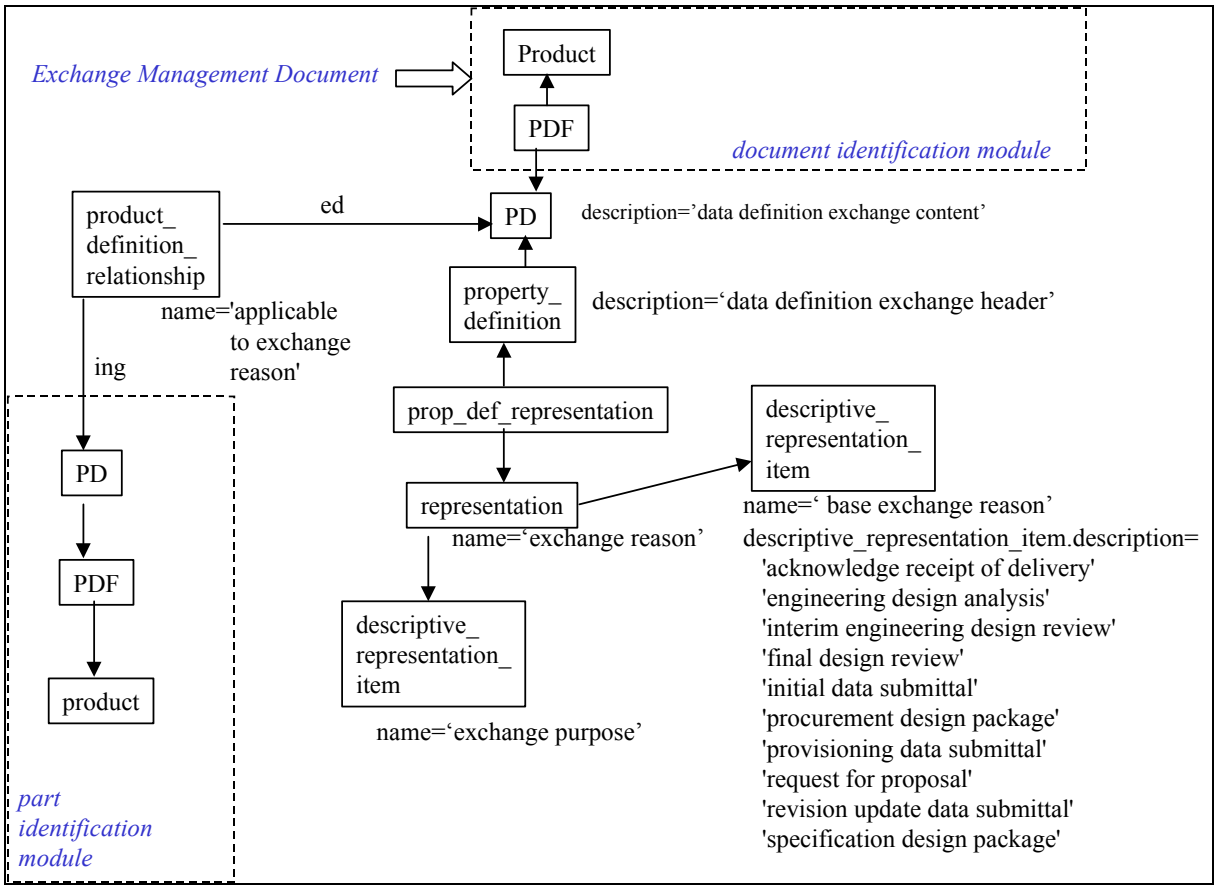


Figure 6.30 - Reason for the Exchange

7 Parts List

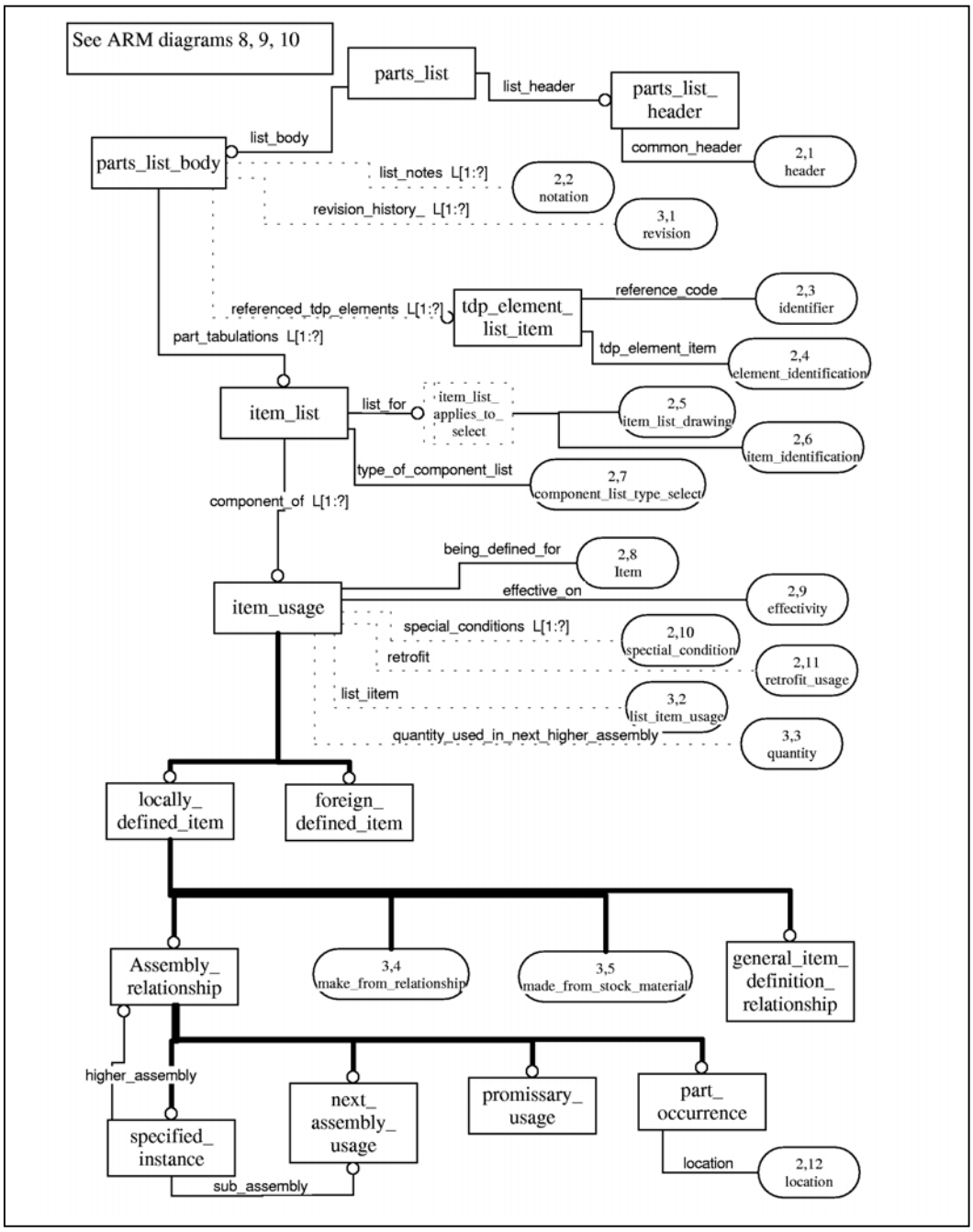


Figure 7.1 - ARM EXPRESS-G for Parts List

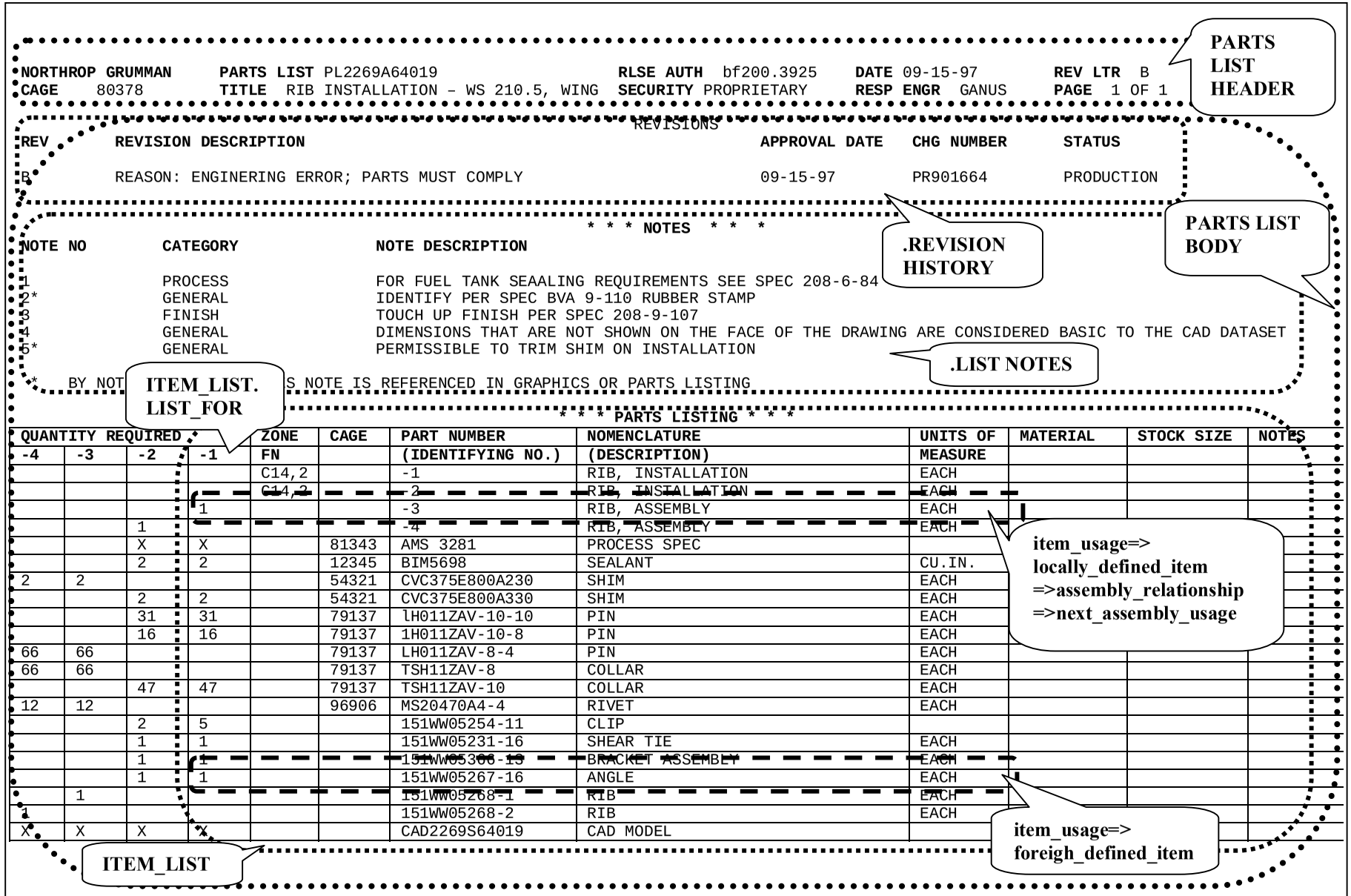


Figure 7.2 Example Parts List with ARM Entities Identified

A Parts list is a type of business product data document that is a configuration controlled tabulation of all parts and bulk materials used in the item (except those materials that support a process). There are different types of parts lists based on the context of its part tabulations. Legacy drawing-based parts lists plus new parts-based parts list are supported. The parts list, just like any other document, binds a collection of data based on some business need. Other information that can be tabulated on a parts list include referenced documents, notes and revision history. This section attempts to add clarification of how the ARM entities as shown in Figure 7.1 are mapped in the Mapping table for parts list and how this mapping can be kept consistent with the PDM Schema Users Guide. The document Parts List can be viewed as giving a defined scope to the exchange of product structure as defined by the PDM Schema. This scope is defined in the ARM of AP232 as being what paper exchanges based on Technical drawings – Item lists ISO 7573-1983 or ANSI Y14.34M-1989, Parts Lists, Data Lists, and Index lists.

7.1 Part List Header

The Parts List Header is the configuration and data management necessary to manage and control the collection of data being identified within a parts list document. The parts list header is existence dependent on the document header module, section 4.1. Figure 7.3 shows the specialization of the document header module to identify a parts list and parts list header. As shown in this figure, the property definition which represents the header does not require it's own **product_definition** but can point at the **product_definition** for the parts list body.

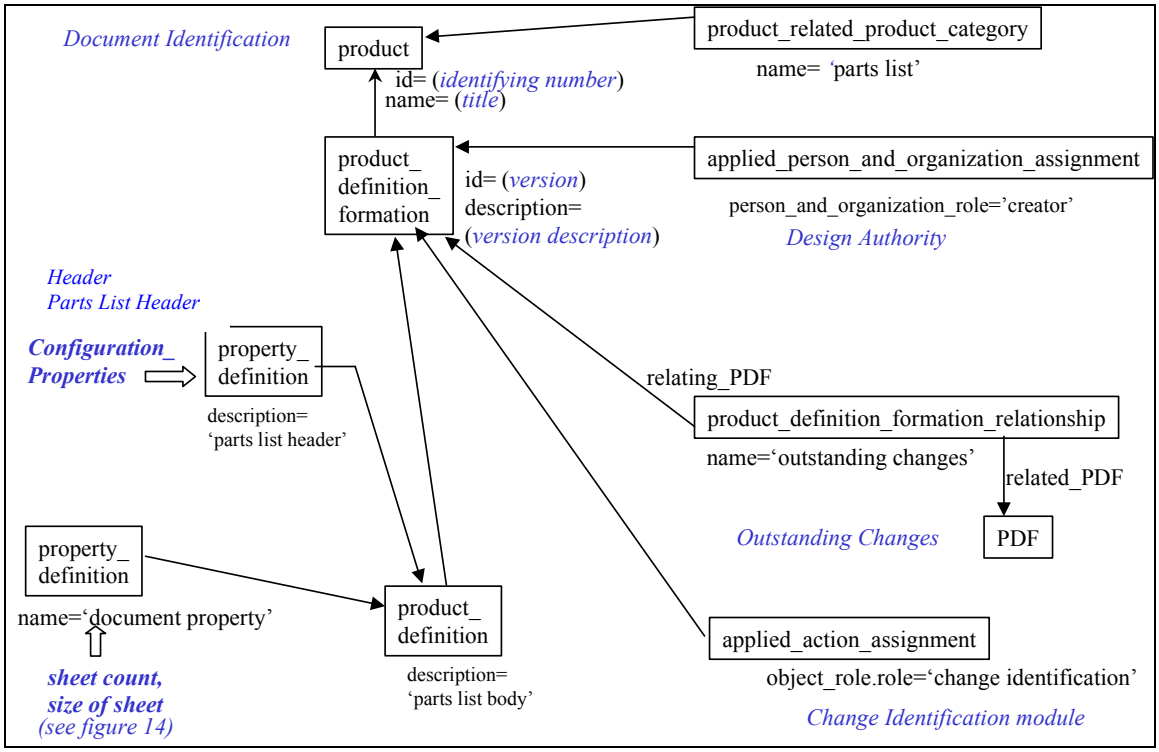


Figure 7.3 - Document Header Specialized for Parts List

7.2 Part List Body

The parts list body is the tabulations of information within a parts list. The tabulated information consists of sufficient identification of Items and related information to meet design disclosure requirements for the Item or Tdp_element defined in the Parts List. The data associated with a Parts List Body are the following:

— Parts Tabulations;

— Revision History;

— List of Notes;

— Reference documents.

Figure 7.4 - Parts List Using Tabulation Based on an Item provides an overview of the AIM entities that need to be instantiated to connect the Parts List Body (a **product_definition**) to the different tabulations of information.

7.2.1 Part Tabulations

A parts list can contain one or more part tabulations or as identifed in the ARM, item_lists. Part tabulation is a list of parts based on a business need. The primary use is to identify what collection of parts goes into an assembly. A part tabulation can identify what part or stock was used to make a detail part. A part tabulation can also provide a list of parts that make a repair kit, an option or an installation drawing. Figure 7.4 and Figure 7.5 provide examples of the AIM entities used to capture these different part tabulations.

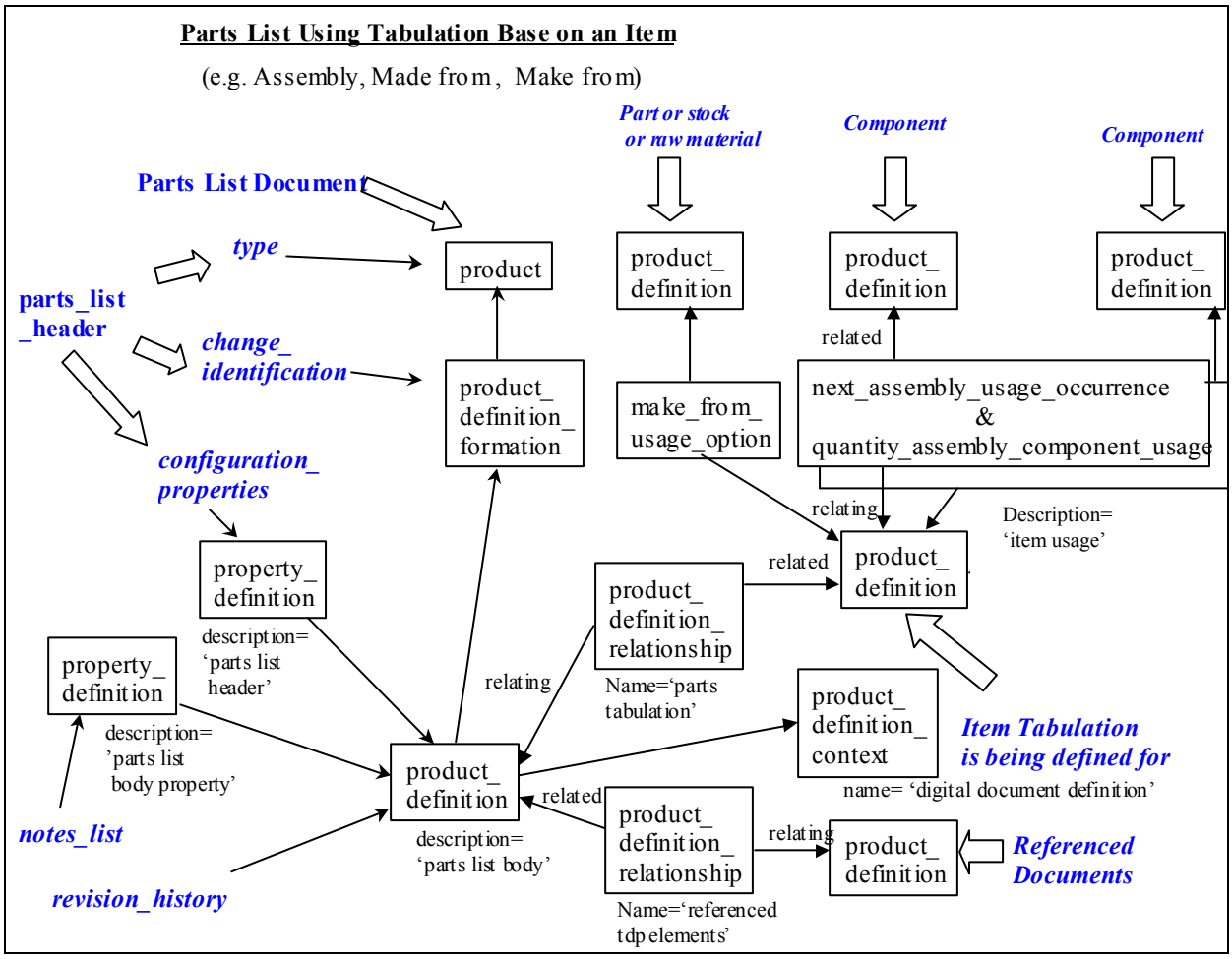


Figure 7.4 - Parts List Using Tabulation Based on an Item

The parts identified in a Part Tabulation may have defining or usage information parameters associated with them. These information parameters are identified in the following sub-sections of 7.2.1.

7.2.1.1 Tabulation List For

The tabulation list for identifies which assembly or document this list will define. The tabulation list for defines the ARM attribue item_list.list_for. There have been three primary concepts that have been identified, based on current practices. There are the following:

— If the tabulation is for a part or assembly and the drawing or document for the part or assembly representation is not known.

— If the tabulation is for a part or assembly and the drawing or document for the part or assembly is known.

— If the tabulation is for a list of parts in a document and there is not a part or assembly in the tabulation.

When the tabulation is for a particular part or assembly, the tabulation (**product_definition**) is of a part, as shown in Figure 7.4. When the tabulation is for a particular document (such as a kit), the tabulation (**product_definition**) is of a document shown in Figure 7.5.

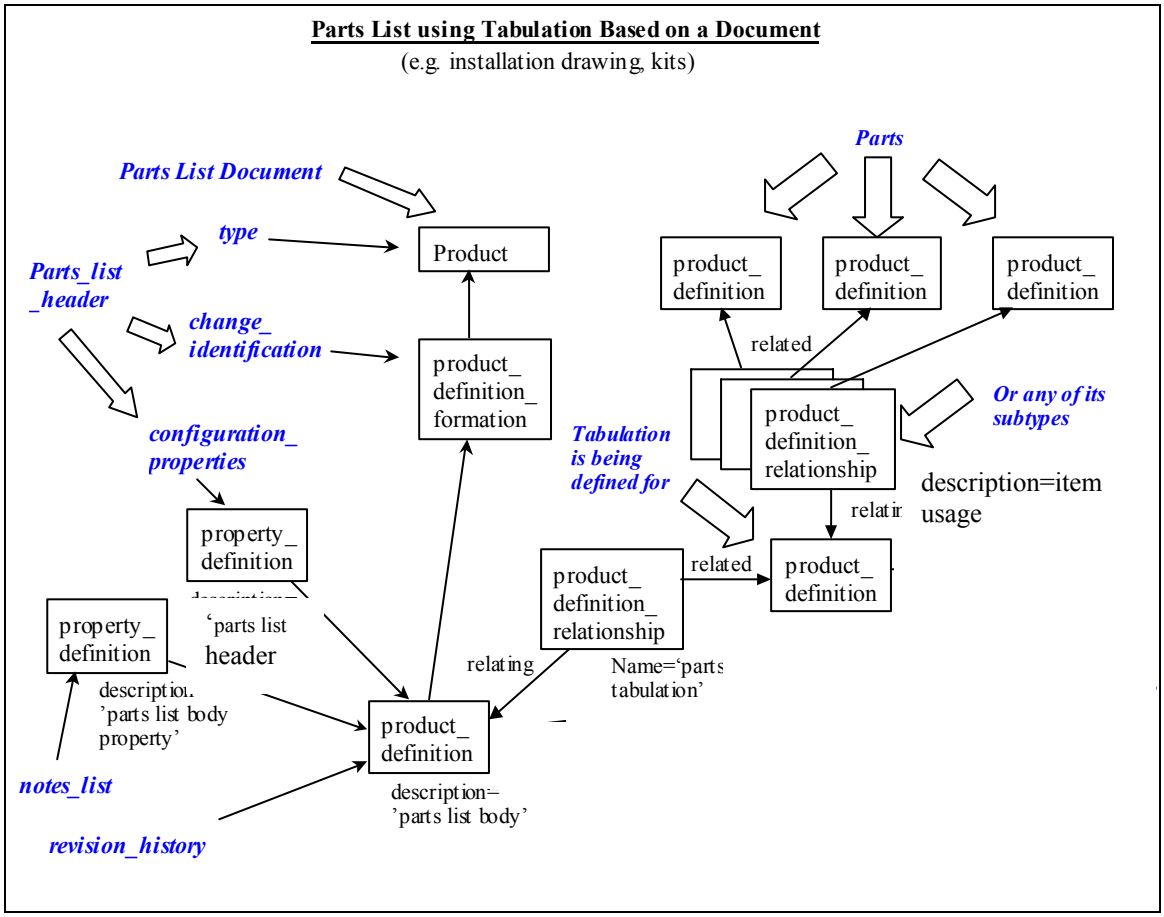


Figure 7.5 - Parts List using Tabulation Based on a Document

For a drawing based parts list, which can consist of multiple part tabulations, each part tabulation (usually an assembly) is associated with the drawing through the use of a **product_-definition_relationship**. The **product_definition_relationship .description** attribute would be constrained to equal 'parts tabulation'.

7.2.1.2 Context of Product_definitions

The context of the product_definitions serving in the role of item_list define mulitple pieces of information. The **product_definition. frame_of_reference** points to a **product_definition_context** which defines the application which defined the parts list, See section 1.1.2.4 of the PDM Schema Users Guide. A **product_definition_context_association** allows association of multiple **product_definition_contexts** to a **product_definition.** For a **product_definition** which is defining a parts tabulation (or in ARM terms item_list) should have an associated **product_definition_context** with a name of 'item list' to identify it.

The identification of the type of component list of each tabulation is captured in **a** third **product_definition_context.** There are some suggested string values for tabulation context:

— ‘assembly defined on drawing component list’;

— ‘made from component list of one’;

— ‘synthetic part number component list’;

— ‘installation drawing component list’;

— ‘item not defined on next higher assembly component list’;

— ‘net change list’.

7.2.1.3 Item Usage

Item usage describes the relationship between the part in the list and the usage of that part in the context of the list. The most prevalent usage is an assembly relationship. There are many parameters that can be recorded about an item’s usage, such as its identity, number needed, and retrofit information.

7.2.1.3.1 Assembly Relationships

The types of assembly relationships are next assembly usage, part occurance, specified instance and promissory usage.

The first type of Locally_defined_item is Next_assembly_usage. The assembly components are related to the assembly through a **next_assembly_usage_component** entity. This entity can be combined into a complex instance with **quantified_assembly_component_usage** to also provide the quantity needed for the assembly. This is shown in Figure 7.5. With next_assembly_usage representations in the parts list, it would be customary to expect that only one instance of **next_assembly_usage_component** for a particular combination of assembly **product_definition** and component **product_definition** would appear in that parts list exchange. An exception could be for split effectivities but the reciever should be notified to expect such a structure. Thus next_assembly_usage is assumed to be mutually exclusive of part_occurance representation.

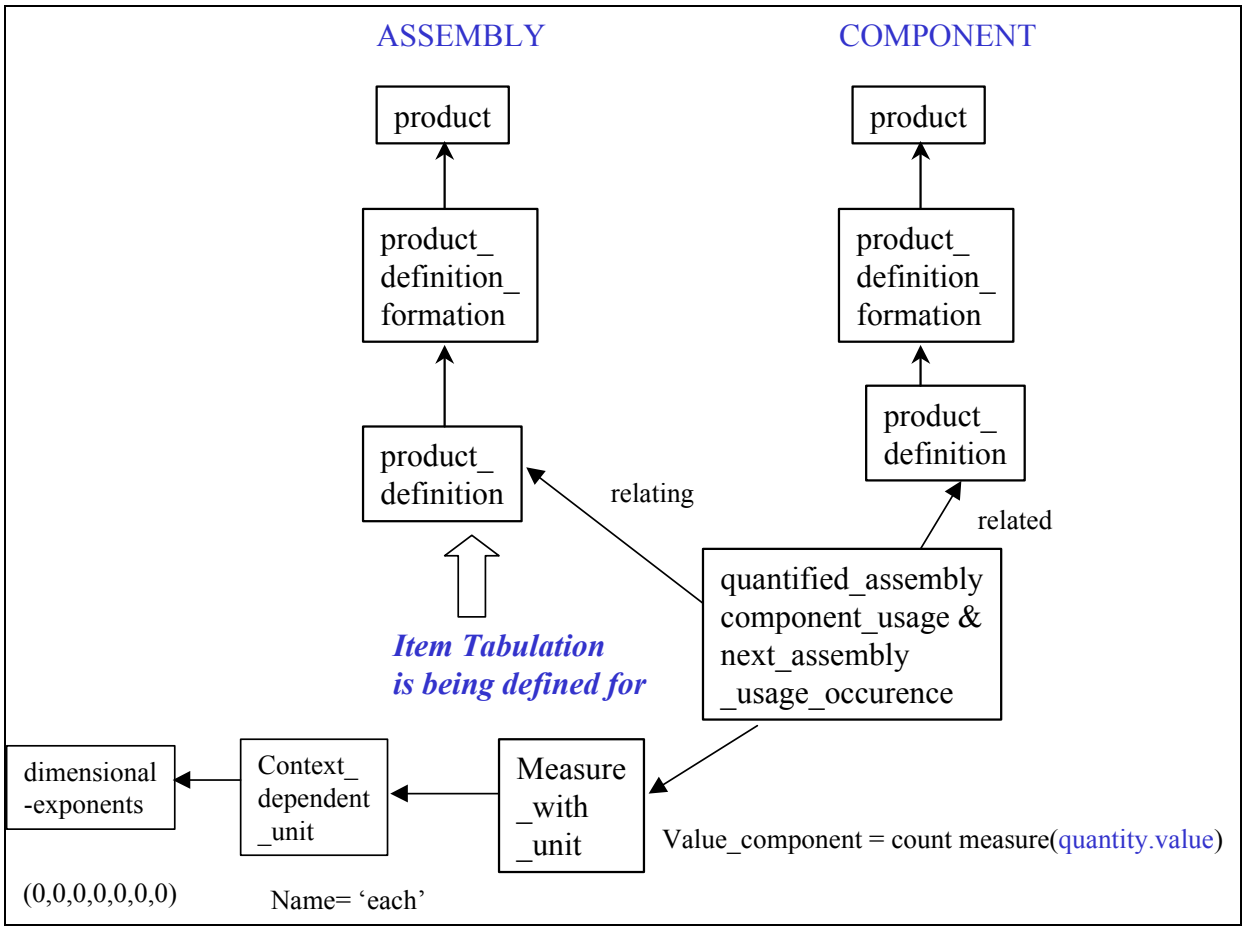


Figure 7.16 - Next_assembly_usage

The part_occurance is used most often for digital mock-up applications where each instance has a defined location. For more information, see Section 4.1.3 Multiple Individual Component Occurences of PDM Schema User Guide. Please note that the optional ARM attribute item_usage.quantity_used_in_next higher_assembly would not be present for usage of part_occurance. As mentioned previously, part_occurance representation is assumed to be mutually exclusive of next_assembly_usage representations. At a minimum, a receiving system should expect different **product_definition**'s with a product_definition_context.name giving the reciever notice of the two representations. An exception would be when digital mock-up applications want to specify common hardware by quantity. Such an exception should be arranged with the receiver ahead of time.

Specified_instance is the capability to identify individual occurrences of component in a muti-level assembly. See section 4.2 Multi-level Assembly Digital Mock Up of the PDM Users Guide for more detailed information.

Promissory_usage is often used early in a design program to allow long lead-time ordering of parts or materials. The use of promissory_usage allows this definition untill the complete product structure is defined and the final next assembly usage is defined. Promissory_usage is mapped to the resource entity **promissory_usage_occurrence**.

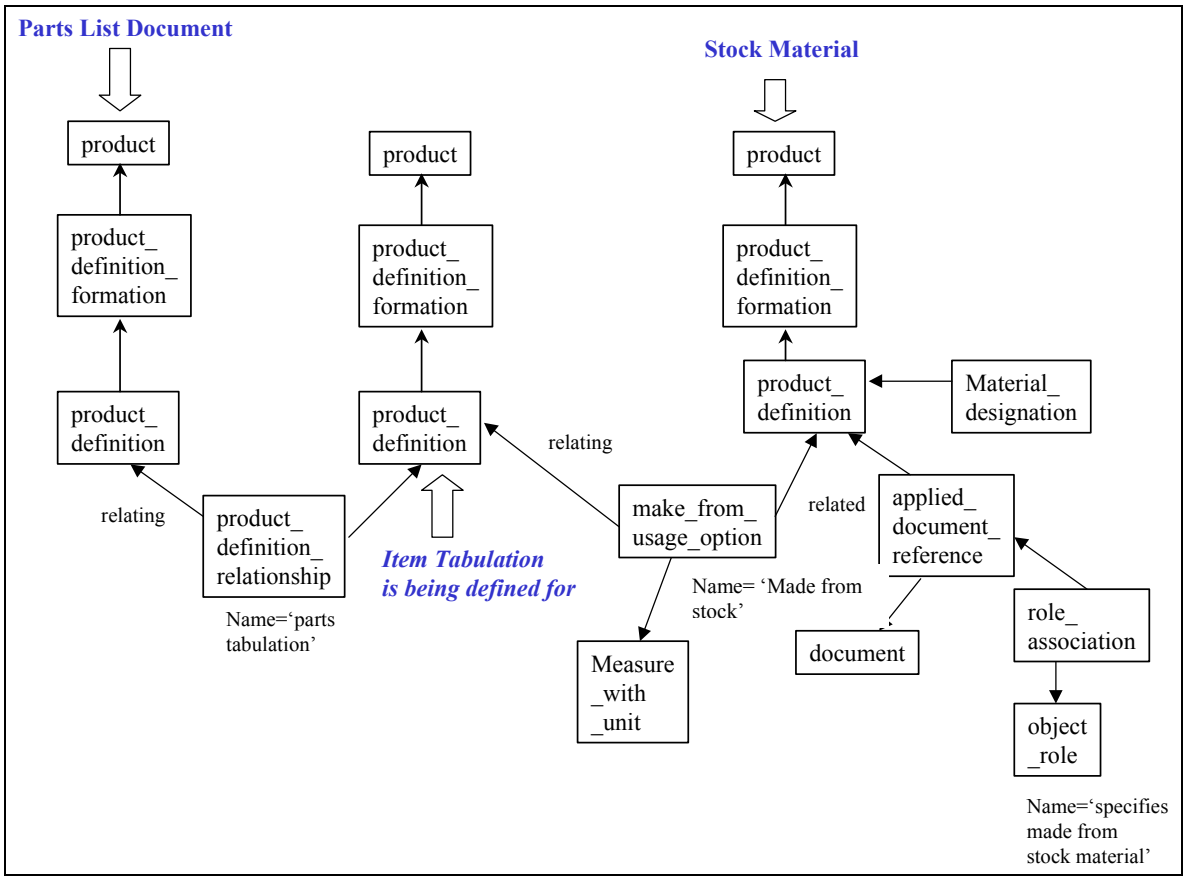


Figure 7.17 - Make From Stock Material

7.2.1.3.2 Part Made from Stock Material

The second type of Locally_defined_item is Make_from_stock_material. Since the design of this item is defined locally, usually within the context of this drawing, the material that is used to manufacture the item is defined. This relationship is mapped to a **make_from_usage_option** with the related attribute defining the material. This material may have a **material_designation**. The **make_from_usage_option** shall have an **applied_document_reference** that specifies the document that identifies the material or process used for the definition of the material stock. The quantity of item made from stock may be defined.

Figure 7.17 shows how this is represented. When a part is made from stock material, this would normally exclude other next_assembly_usage relationships. That is to say that you cannot make something out of stock material and assemble it from components. If **make_from_usage_option** objects are combined with **next-assembly_usage_occurrence** objects, special arrangements should be made with the receiveing systems to make sure that they can handle this case. Such a case might be the example where the sending systems calls out explicitly finish materials such as paint.

7.2.1.3.2.1 Stock Material

A stock material is raw material that material vendors supply for multiple uses. Stock material usually comes in standard sizes based on material type. Thus simple shapes, such as plate or bar, may be defined by a stock size while more complicated shapes may be separate items with their

own physical definition. AP232 defines many of these in the ARM entity stock_size_parameters. The standard shapes are mapped to **descriptive_representation_item.description**.

7.2.1.3.3 Part Made from Another Part

The third type of Locally_defined_item is Make_from_relationship. In this case, a part is locally modified to become the new part. The most common scenario is when a purchased part or assembly is modified for an application. This can define the case where multiple items are made out of the part which is acting as stock.

Make_from_relationship is mapped to a **make_from_usage_option**. The quantity attribute points to a **measure_with_unit** that defines the quantity. The related_product_definition attribute points to the part acting as stock. Again, just like with made_from_stock_material, the presence of a **make_from_usage_option.relating_product_definition** would normally exclude the presence of a **next_assembly_usage_occurrence.relating_product_definition.relating_product_definition** pointing to the same **product_definition.** To maintain compatibility to AP203, an optional mapping is to **design_make_from_relationship**. This mapping is limited to cases where one part is made from another part and the new part's design is derived from the original part.

7.2.1.3.4 General item definition relationship

General item definition relationship is the more general case of make from relationship. It should only be used where the semantics of make_from_relationship does not apply. An example would be when the only the design of this item is derived from another item. This maps to a **product_definition_relationship** where the description attribute equals 'replacement definition' or 'derivation'.

7.2.1.3.5 Foreign Defined Item

A foreign defined item is defined on another drawing from this drawing. A foreign defined item is mapped the same as next_assembly_usage. The **product_definition_relationship** to the drawing where it is assembled has a **product_definition_relationship.name** = 'foreign defined'. See
Figure 7.15.

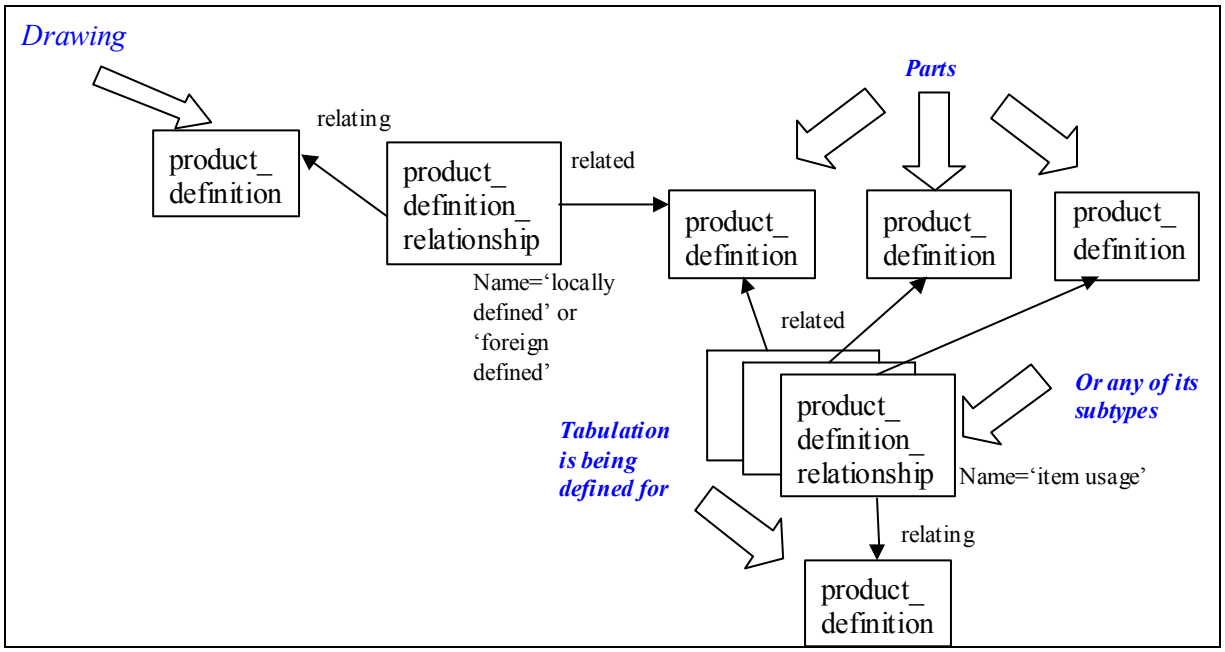


Figure 7.15 - Locally Defined or Foreign Defined

7.2.1.4 Part Assembly Quantities

Part quantities for an assembly have three parameters that are captured. The parameters are the quantity value, the quantity units, and the quantity accuracy. The quantity value and quantity units map to a **measure_with_unit**. The quantity accuracy maps to **measure_qualification**. Figure 7.18 shows these two mappings, respectively. See Figure 7.14 for another version of **measure_with_unit**.

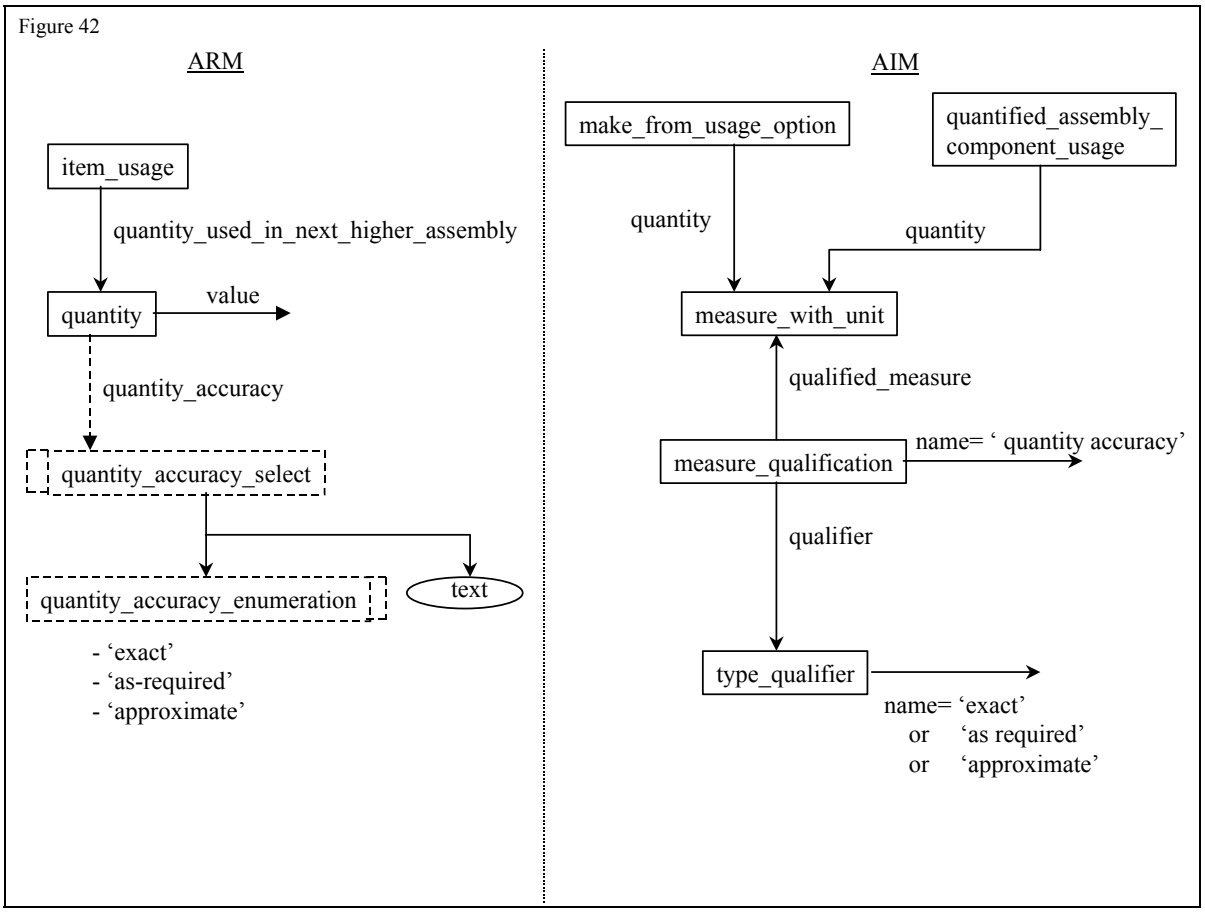


Figure 7.18 - Quantity Accuracy

7.2.1.5 Retrofit Usage

The retrofit usage specifies the definition and disposition of an item that is replacing another item that is a component of a product. The replacing Item may be the original Item that has been modified. The data associated with the Retrofit Usage are the following:

— defining document;

— disposition for;

— retrofit description.

The retrofit description specifies a characterization of the retrofit. The defining document specifies a document that defines the actions required for the retrofit. The disposition_for specifies the action to be taken with the original item. The value of the disposition for shall be one of the following:

— **add**: add specifies that the Item was added as a result of the retrofit.

— **delete**: delete specifies that the Item was deleted as a result of the retrofit.

— **modify**: modify specifies that the Item was modified as a result of the retrofit.

Figure 7.9 provides the resultant mapping for the retrofit usage data.

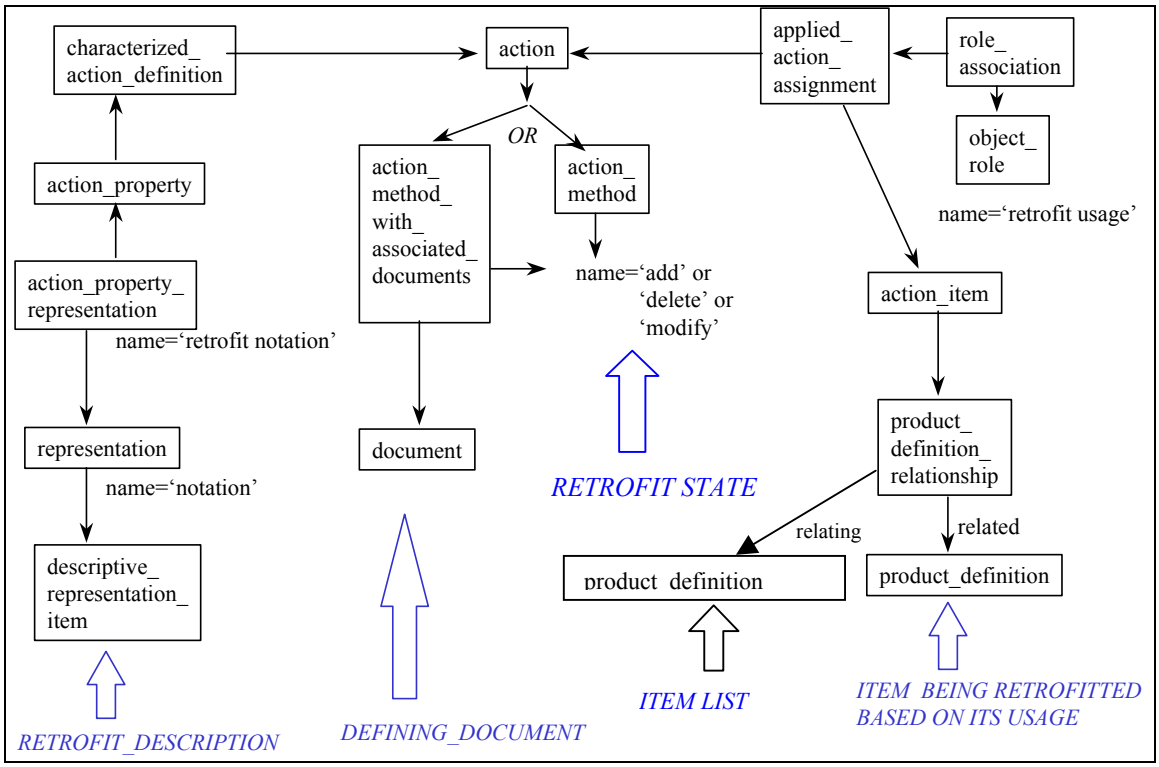


Figure 7.9 - Retrofit Usage

7.2.1.6 Usage Effectivity

Usage effectivity is the ARM effectivity which applies to a item_usage. This maps to a complex entity of **effectivity, configuration_effectivity, product_definition_effectivity,** and one of the effectivity subtypes of **serial_numbered_effectivity, lot_effectivity, dated_effectivity,** or **time_interval_based_effectivity.** This requires that **configuration_item, product_concept, product_concept_context,** and **configuration_design** entities be instantiated. The **configuration_design.design** attribute must point to either a **product_definition** or a **product_definition_formation.** Although the semantics of this attribute do not make it required, in most cases the **product_definition** or **product_definition_formation** should be of the end item product of this facility for the **configuration_item. Product_definition_formation** should be used unless there are specific semantics require view dependency then use **product_definition. Applied_effectivity_assignment** may be used instead of **configuration_effectivity** and **product_definition_effectivity** in the specific case where the item_list is for a drawing and the next higher part is not known. **Effectivity** may optionally have an **approval,** a name via a **named_attribute,** a work activity via a **applied_activity_assignment,** and a description via a **description_attribute.**

7.2.1.7 Special Condition applied to Part Usage

See 5.4 Special Condition for detail discusssion. The **property_definition_representation** which connects the **representation** should have it's **definition**attribute pointing to the **product_definition_relationship** of the item_list.

7.2.1.8 List Item Usage Parameters

List item usage is a group of optional values that are defined for each entry item_usage. These values are drawing and configuration information. What is unique is that there are values that apply for this usage. See figure for the ARM definition.

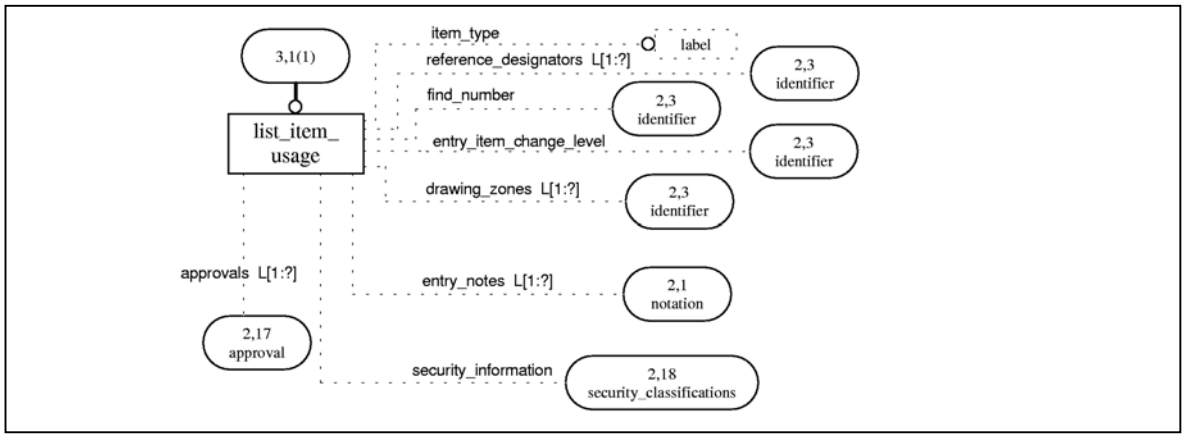


Figure 7.10 - ARM diagram for list_usage _item

The data associated with a List_item_usage are the following:

— Reference Designators;

— Find number;

— Entry notes;

— Drawing notes;

— Entry Change Level;

— Item Type;

7.2.1.8.1 Reference Designators

Reference designators are a unique identification specified on the drawing and the assembly for each specific instance of a component. Typically reference designators are defined for electronic applications and follow standards or drafting practices in their assignment. There may be zero, one, or many reference designators for each item_usage. If it is specified, the number of reference designators will normally be the same as the quantity.n.

The reference designator is directly mapped to the "reference_designator" attribute of **Assembly_Component_Usage**. When a complex instance of **Assembly_Component_Usage** and **Quantified_Assembly_Component_Usage** is used, the reference_designator may be a comma-

separated list. When a Reference Designator is required with **Make_From_Usage_Option**, AP232 has created the new entity, **Make_From_Usage_Option_With_Reference_Designator**.

7.2.1.8.2 Find Number

The find number is an identifying number on the field of a drawing. The find_number is used in lieu of the part number on the field of the drawing.

The find number is mapped to the "id" attribute of the **product_definition_relationship** between the component **product_definition** and the assembly drawing **product_definition**. See Figure 7.11. This differs from the PDM Schema User's Guide, which maps it to the **product_-definition_relationship** that represents the item_usage. This departure is done to capture the exact semantics of a drawing find number and allow use of the other "id" attribute to be used for uniqueness of this attribute for split effectivities used in some applications.

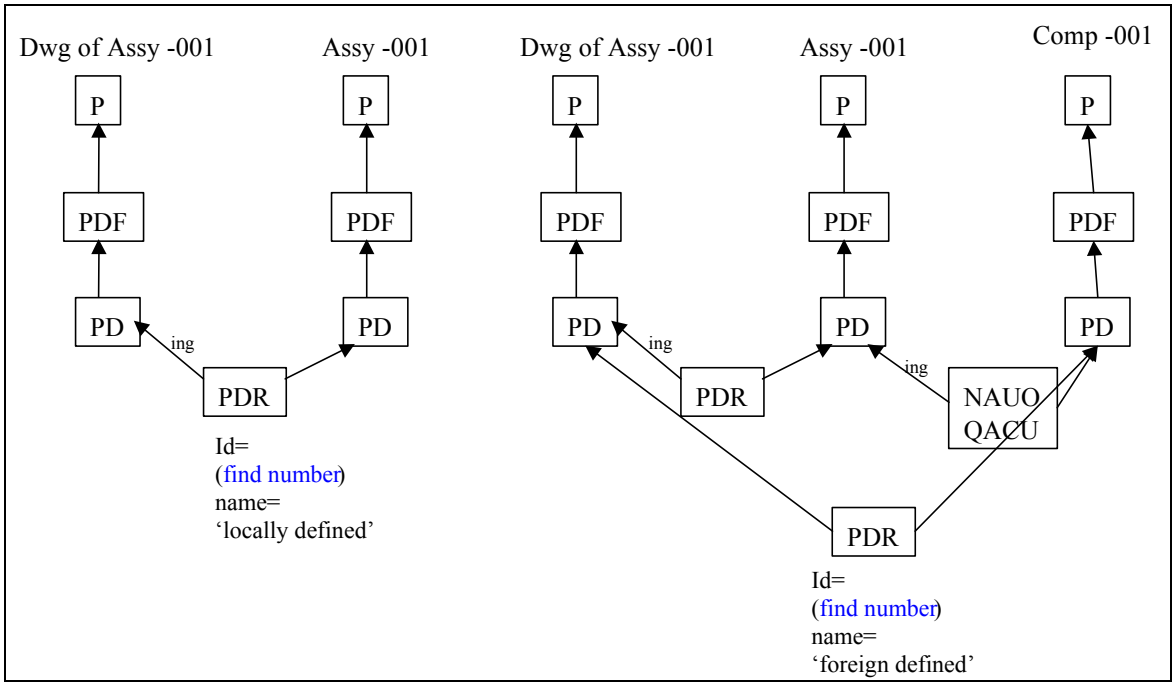


Figure 7.11 - Find Number

7.2.1.8.3 Entry notes

Notations applicable to this particular item_usage are entry notes. Notation is represented as **descriptive_representaion_item**'s just as defined in 5.1. The major difference is that the **property_definition.definition** attribute points to **product_definition_relationship** (or its subtype) which represents the item_usage.

In some presentations of parts lists, these Entry Notes would appear as note numbers, and the actual note text would appear in the document notes. For consistency, it is suggested that these **descriptive_representation_item**s only be linked to the **product_definition_relationship** and not be linked to the overall **product_definition**.

7.2.1.8.4 Drawing Zone

Drawing zones normally specify the location of the component callout in coordinates on the drawing sheets. This is an aid in locating the geometrical representation in large complex multi-sheet drawings. This standard does not define how these Drawing Zones are defined. A format that could be used is sheet number, comma, drawing zone. Thus "3,B4" would say drawing sheet 3, coordinate B in one direction and coordinate 4 in the other direction. Any other format of this data would need to be agreed upon in a business agreement prior to the exchange.

Drawing zones are represented in **descriptive_representation_item** entities where the **representation_item.name** and **representation.name** attributes are 'drawing zone.' See Figure 7.12.

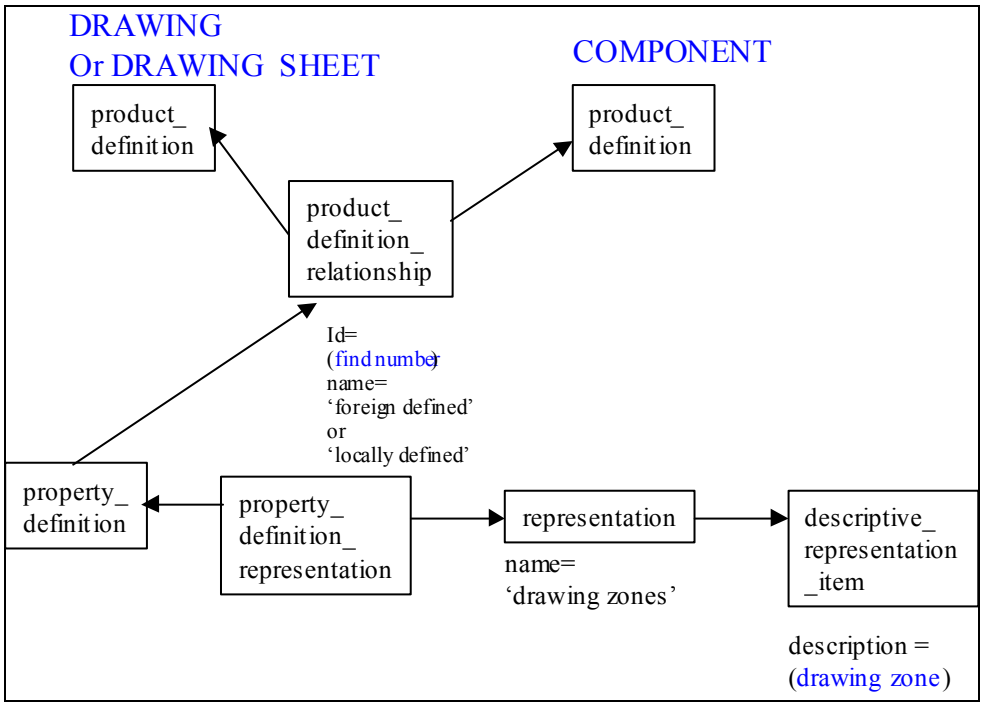


Figure 7.12 - Drawing Zones

7.2.1.8.5 Entry Item Change Level

A change level defined for the entry item in this exchange is to be interpreted to be a change level defined for this entry item. See Section 4.2.

7.2.1.8.6 Item Type

The item type specifies a label that identifies the classification of the Item during a particular use of the item. This item type is defined as a product category. A product category defined against an item in this exchange, which is a list_entry, is assumed to be specifying an iItem type for this list entry.

7.2.2 Revision History

Revision history is covered in section 5.2. The Revision history is related to the **product_definition** for the parts_list_body by an **applied_action_assignment.** This **applied_action_assignment** should have a **role_association** of **object_role.name** equal to 'revision history'.

7.2.3 List of Notes

The list of notes for a parts list is the total of all notation as defined in section 5.1 that are associated to the **product_definition** of the parts list body by a **property_definition_-representation** and a **property_definition.**

7.2.4 Referenced Documents

Referenced documents are related to the **product_definition** of the parts list body by a **production_definition_relationship** with a name attribute equals to 'referenced Tdp_elements'. The **product_definition_relationship.related_product_definition** will identify the **product_-definition** of the referenced document. This is consistent with the PDM Schema Usage Guide as defined in section 11.2.

7.2.5 Item

An item is any unit or product including parts, assemblies, equipment, accessories, or attachments.
An iItem is represented by a product_definition.

7.2.5.1 Item Identification

An item identification is the identification of an Item for purposes of configuration control. Since an Item requires a design authority, the identification of an item, as a minimum, includes the part number and the company who created or owns the part design.

The data associated with an item identification are the following:

— alternate identifications;

— change;

— classifications;

— design activity;

— identifying number;

— nomenclature or name;

— source information.

These are shown in Figure 7.13.

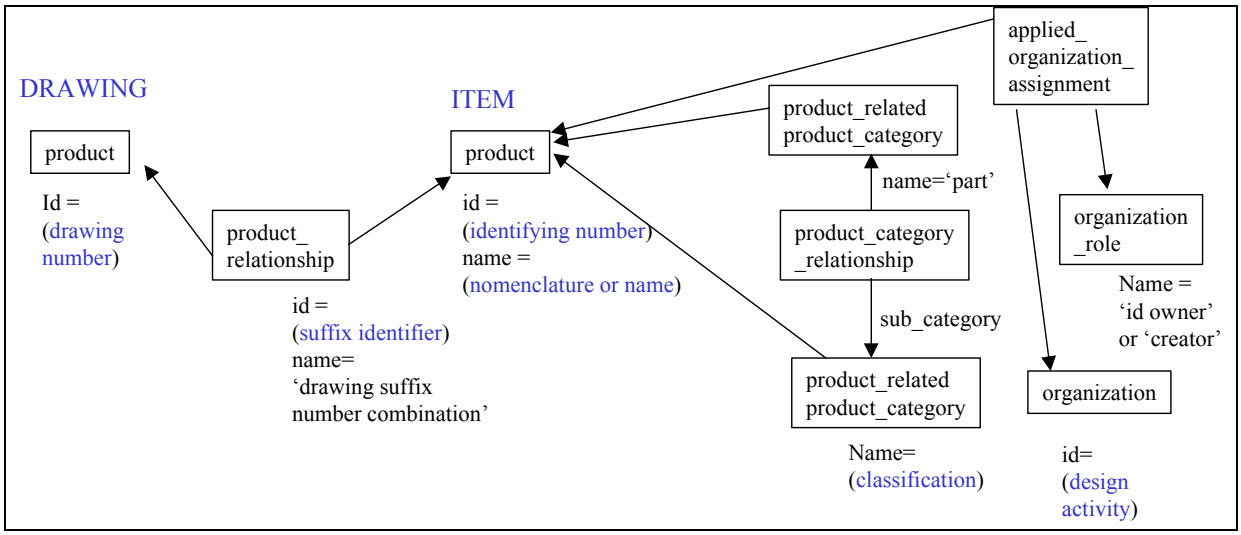


Figure 7.13 - Item Identification

7.2.5.2 Item Configuration

The item configuration specifies configuration control information pertaining to the item. This topic is discussed in section 4.3.

7.2.5.3 Alternate Item

The alternate item specifies a list of substitute Items that meet the form, fit, and function of the design and may be used as an alternate to the original Item. An alternate may be a substitute with no concern for which assembly context, in which case it maps to an **alternate_product_-relationship**. An alternate which is only effective in a specific assembly context is mapped to an **assembly_component_usage_substitute**. The alternates need not be specified for a particular Item. There may be more than one alternate for an Item.

7.2.5.4 Weight

The item_weight specifies the heaviness of the item. This is shown in Figure 7.14.

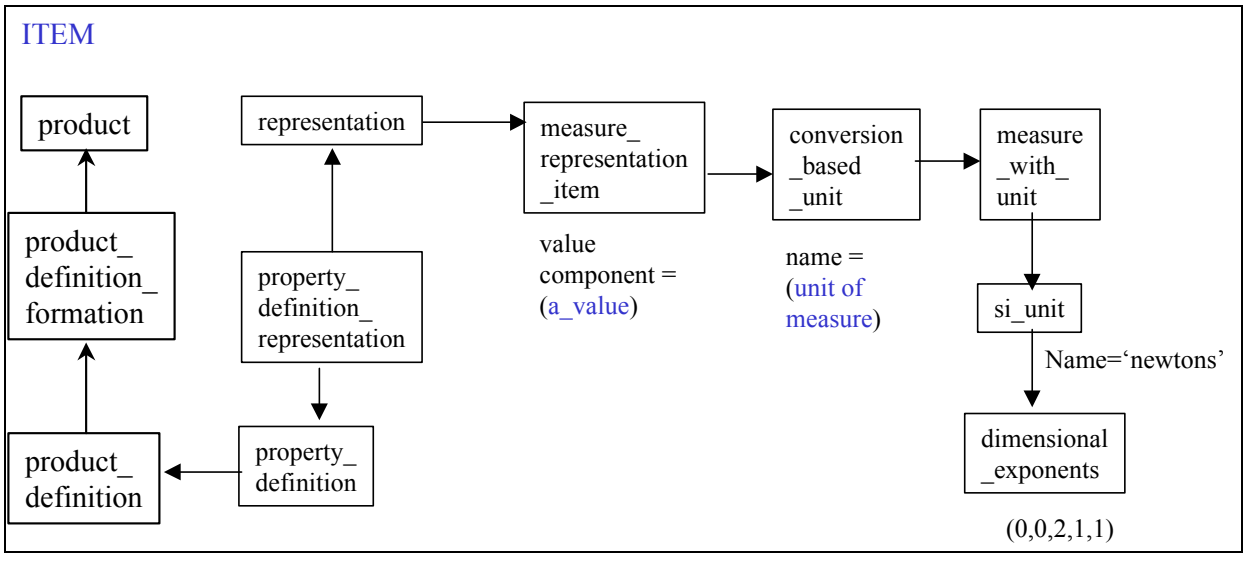


Figure 7.14 - Weight of an Item

7.2.5.5 Item Contexts

Item context specifies context information pertaining to the item. The iItem context need not be specified for a particular Item. There may be more than one item context for an item.

The contexts are specified as **product_definition_context.name**. This is identified by the **product_definition.frame_of_reference attribute**. Since item contexts may vary considerably between trading partners, it is suggested that item contexts be identified in business agreement prior to exchanges.

7.2.5.6 Special Conditions

Special conditions specifies mutually agreed upon conditions that apply to the item. The mutually agreed upon conditions relate to codes or designations for the Item. Special Conditions need not be specified for a particular Item. There may be more than one special conditions for an Item.

Special conditions are defined by **descriptive_representation_item**s. These are defined similar to entry characteristics as shown in Figure 6.8 - Top Nodes Identified in List for Exchange Management

. Since special conditions may vary considerably between trading partners, it is suggested that special conditions be identified in business agreement prior to exchanges.

7.2.5.7 Conditions Defined through simple reference

The conditions defined through simple reference specifies any type of condition of an Item that can be obtained through a simple reference to a document. The conditions defined through simpl reference need not be specified for a particular Item. There may be more than one conditions defined through simple reference for an Item.

EXAMPLE 1 Final conditions such as final material or process characteristics.

EXAMPLE 2 Finish conditions such as surface finish or external characteristics (e.g. paint).

EXAMPLE 3 Design specifications.

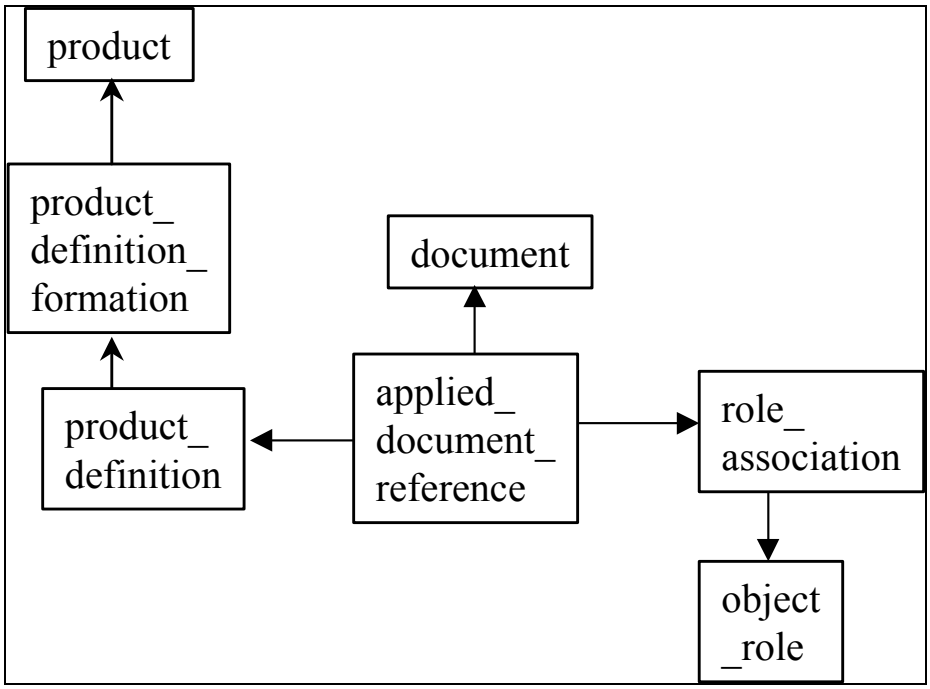


Figure 7.15 - Conditions Defined Through A Simple Reference

Conditions defined through a simple reference is shown in Figure 7.15.

7.2.5.8 Conditions Defined through Constrained Document

The conditions defined through constrained document specifies any type of condition of an Item that can be obtained through the use of constraining the use of a document. Constraining parameters are used to aid in defining the condition of the item. The conditions defined through constrained document need not be specified for a particular item. There may be more than one conditions defined through constrained document for an item.

EXAMPLE 1 Final conditions such as final material or process characteristics.

EXAMPLE 2 Finish conditions such as surface finish or external characteristics (e.g. paint).

EXAMPLE 3 Design specifications.

Conditions defined though a Constrained Document is shown in .

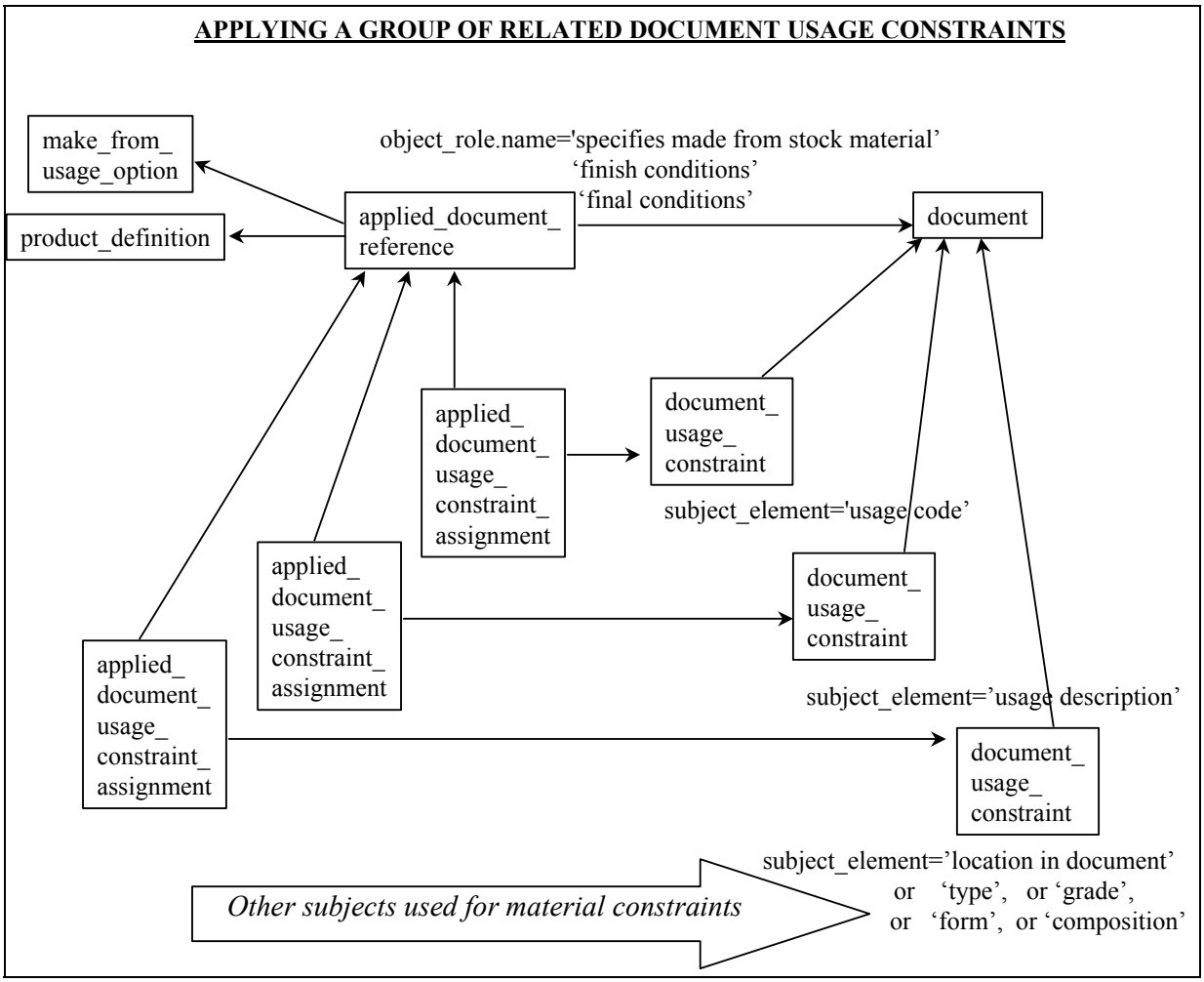


Figure 7.16 - Conditions Defined Though Constrained Document

7.3 EXPRESS Rules for Parts List

The rules defined in the common table and reviewed in section 5.5 apply to parts list. In addition the following rules are defined.

7.3.1 Dependent_instantiable_named_unit

The **dependent_instantiable_named unit rule** spcifies that a **nameunit** may not be instantiated withour being related to another entity. See Figure 7.16 - Conditions Defined Though Constrained Document.

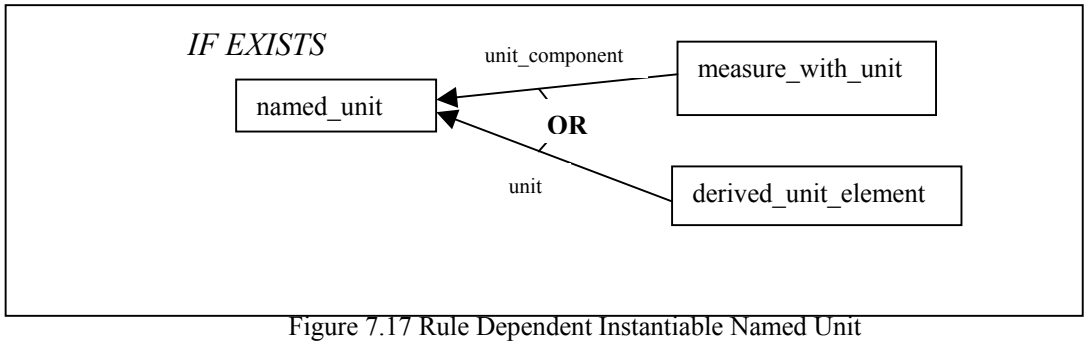


Figure 7.17 Rule Dependent Instantiable Named Unit

7.3.2 Document product equivalence existence rule

The document product equivalence rule existence rule specifies that when a document references other product data through a **document_reference** or **document_usage_constraint** then other entitities must be instantiated. See figure 7.35. This figure show one of six cases, **applied_document_reference,** can be replaced by **applied_document_usage_constraint_-assignment** and the **document_product_equivalence.related_product** may optionally point at a **product_definition_formation** or **product_definition.**

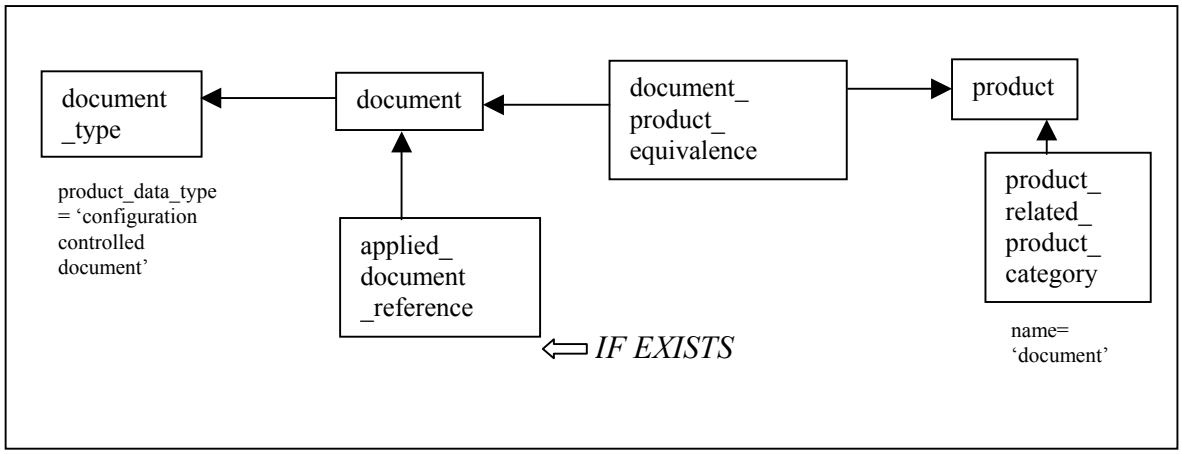


Figure 7.18 - Rule document product equivalence existence rule

7.3.3 drawing suffix number combination identification constraint

The drawing suffix number combination identification constraint specifies how to identify the document in a drawing sufix number combination relationship.

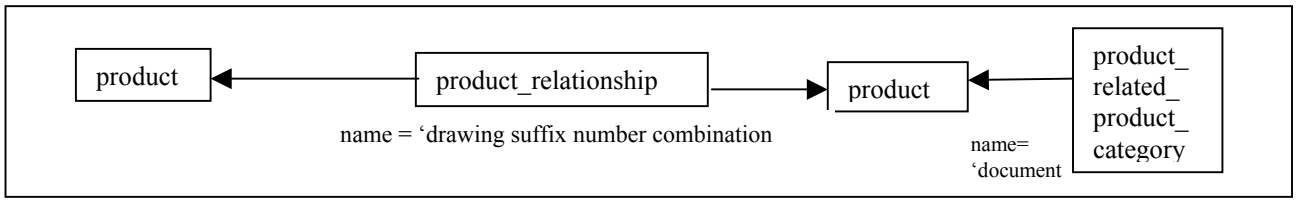


Figure 7.19 - Rule Drawing suffix number combination identification constraint

7.3.4 Identification of sheet constraint

The identification of sheet constraint rule identifies the **product_related_product_categories.-name** which identify a sheet.

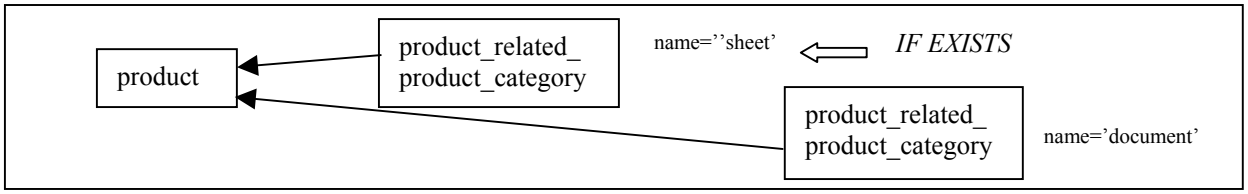


Figure 7.20 – Rule Identification of sheet constraint

7.3.5 Item source information identification constraint

The **item source information identification constraint** rule specifies the string values that identify the item scource information

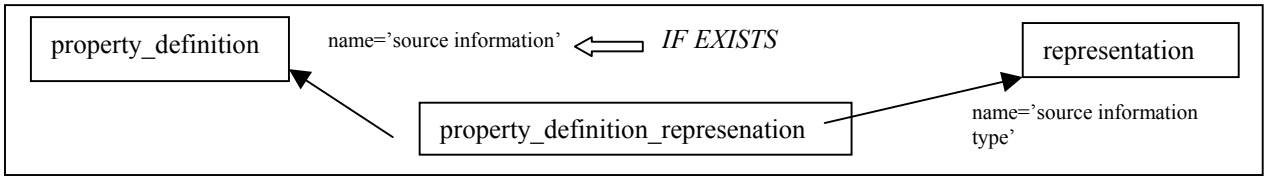


Figure 7.21 – Rule item source information identification constraint

7.3.6 Notation type identification constraint

The **notation type identification constraint** rule specifies the string values that identify notation.

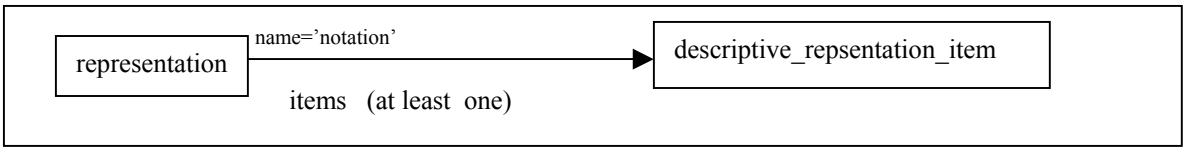


Figure 7.22 – notation type identification constraint

7.4 Informal Rules

The following rules are not formal rules but are suggested to maintain compatibility:

— Associated_list to drawing relationship identification constraint;

— Drawing zone identification constraint;

— Foreign or locally defined item identification constraint;

— Parts list requires header;

— Retrofit usage identification constraint;

— Revision approval identification and person organization identification constraint;

— Revision authorizing identification_constraint;

— Revision requires date or date_time.

7.4.1 Associated list to drawing relationship identification constraint

The associated list to drawing relationship identification constraint rule specifies the string value in a **product_definition_formation_relationship** that identifies a relationship between a drawing and its associated lists.

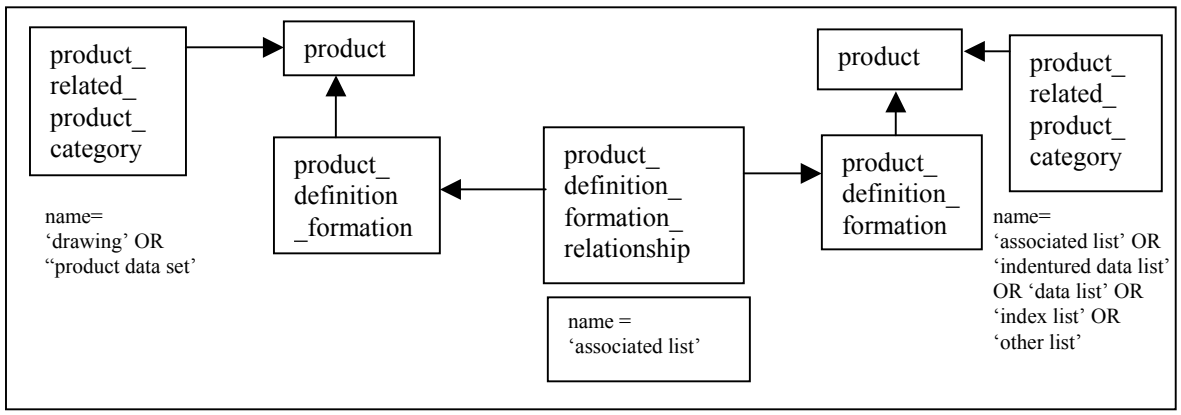


Figure 7.23 Rule Associated list to drawing relationship identification constraint

7.4.2 Drawing zone identification constraint

The drawing zone identification constraint rule specifies the string values that identify how drawing zones are associated to a part.

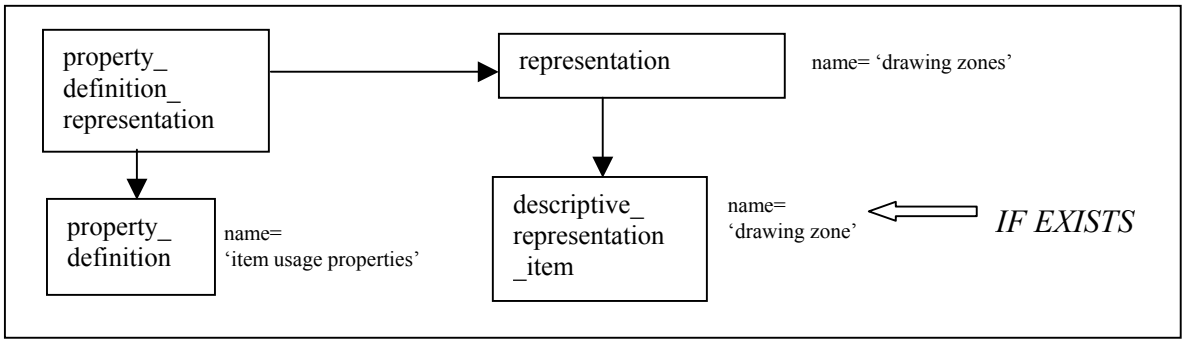


Figure 7.24 Rule Drawing zone identification constraint

7.4.3 Foreign Or Locally defined item identification constraint

The foreign or locally defined item identification constraint rule specifies string values that identifying a part is only referenced on a drawing and not defined on that drawing or is identified as being defined on a particular drawing. This rule is illustrated in Figure 7.15 - Locally Defined or Foreign Defined.

7.4.4 Parts list requires header

The parts list requires header rule specifies that the **product_definition** of the part list document must have a **property_definition** with a description attribute equal to 'parts list header'. This is an important rule in allowing a reciever system to know that this parts list is defined in this file and not just referenced. This rule is illustrated in Figure 7.3.

7.4.5 Retrofit usage identification constraint

The retrofit usage identification constraint rule specifies the string value that identifies the retrofit usage action. The **applied_action_assignment** should have an **object_role.name** equal to 'retrofit usage'. This is shown in Figure 7.9 - Retrofit Usage.

7.4.6 Revision approval identification and person organization identification constraint

The revision approval identification and person organization identification constraint specifies the string values that identify the related approval and association of an organization or person and organization as an approval role to a revision idenfication.

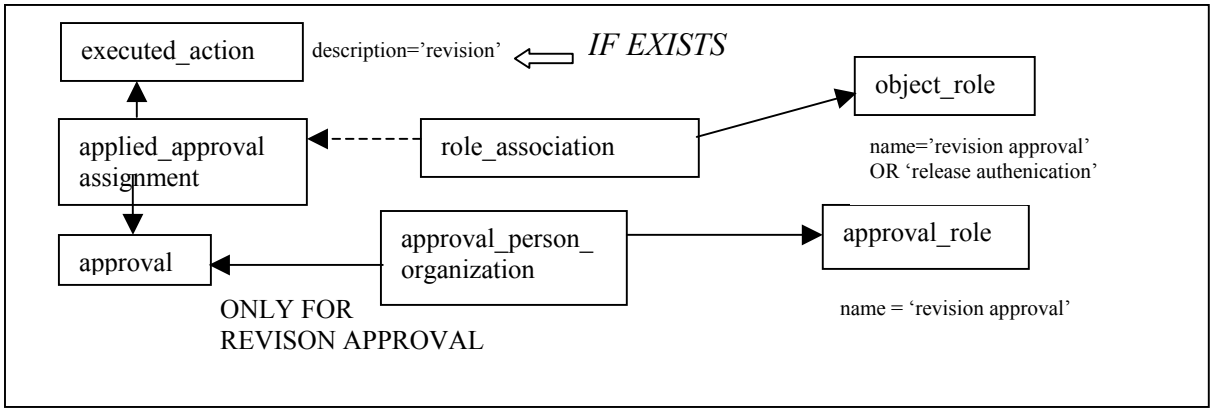


Figure 7.25 - Revision approval identification constraint

7.4.7 Revision authorizing identification constraint

The revision authorizing identification constraint rule specifies the string values that identify the relationship between a revision oa a document and the document that authorized its revision.

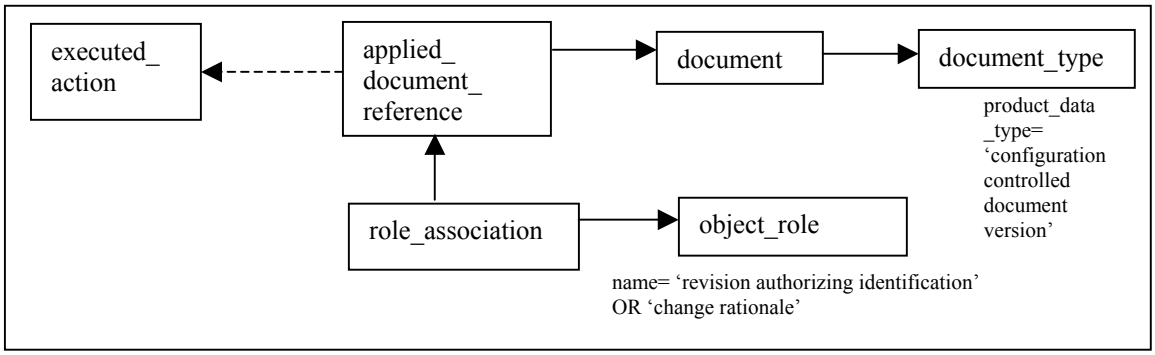


Figure 7.26 - Rule Revision authorizing identification constraint

7.4.8 Revision requires date or date time

The **revision requires date or date time** rule specifies that a Revision shall have an associated date or date and time.

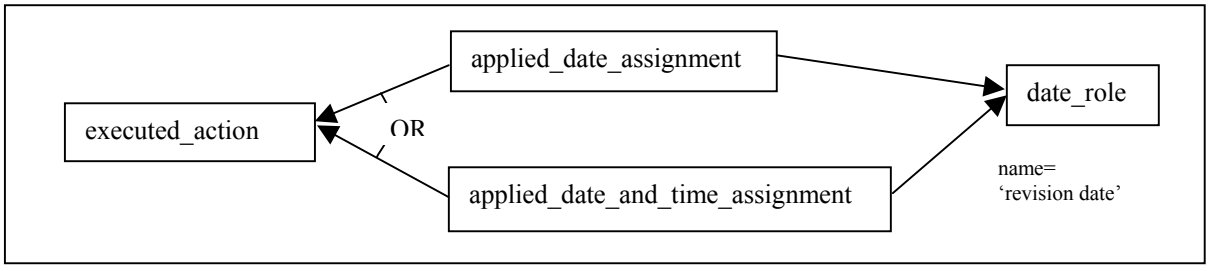


Figure 7.27 –Revision requires date or date time

8 Indentured Data List

An indentured data list is a type of document that has the contents of the data organized in a hierarchical structure. The hierarchical structure is defined as a "top-down or breakdown" order. The relationships among the different types of products, such as parts and documents, provide the indentured structure. An example of an Indentured_data_list is a tabulation of all drawings, referenced documents, associated lists, and other essential in-house documents required for design disclosure pertaining to the Tdp_element or Item to which Indentured_data_list applies.

Figure 8.1 provides an overview of the how an Indentured_data_list (IDL) is put together in STEP. A indentured data list is a type of document that is mapped as a **product.** An indentured data list has header information that captures information such as change identification and configuration properties. The body of the indentured data list is where the content of a document is captured. The body of the IDL document may contain the following:

— list of notes;

— revision history;

— list of standard documents;

— indentured hierarchy of parts and documents.

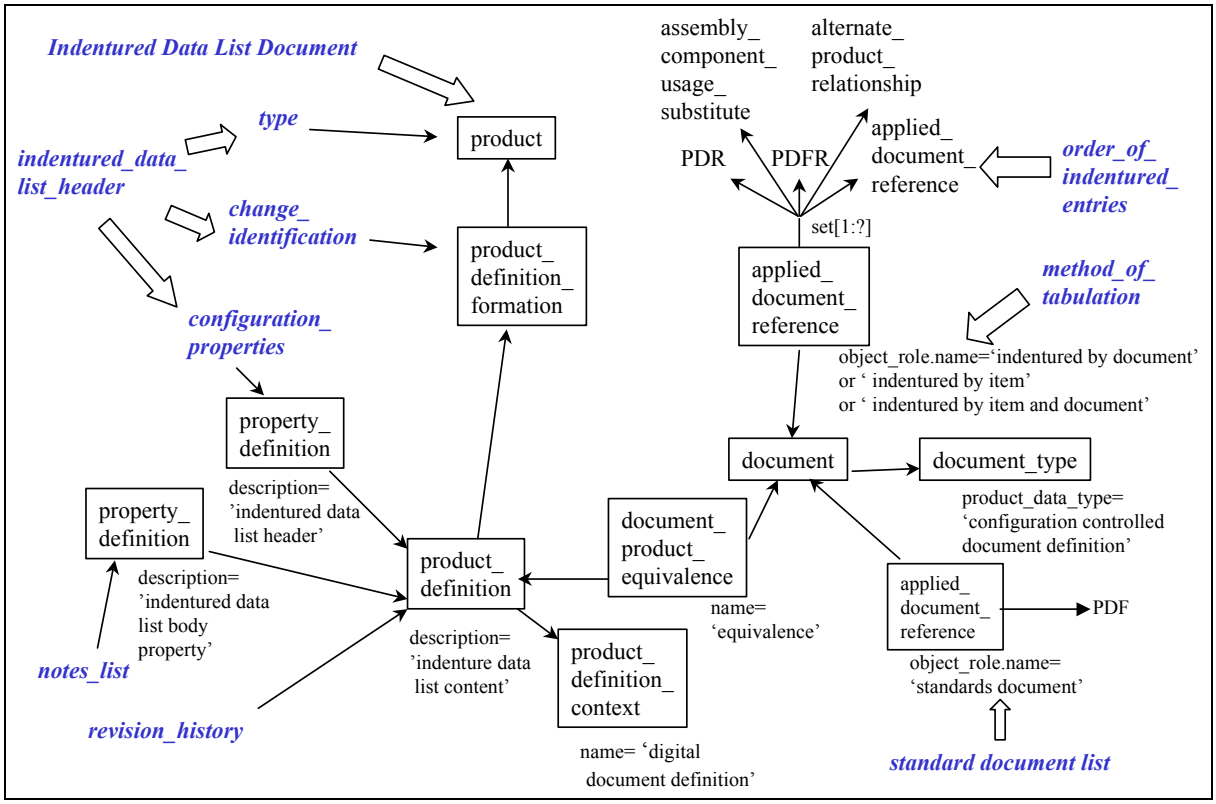


Figure 8.1 - Indentured Data List Module

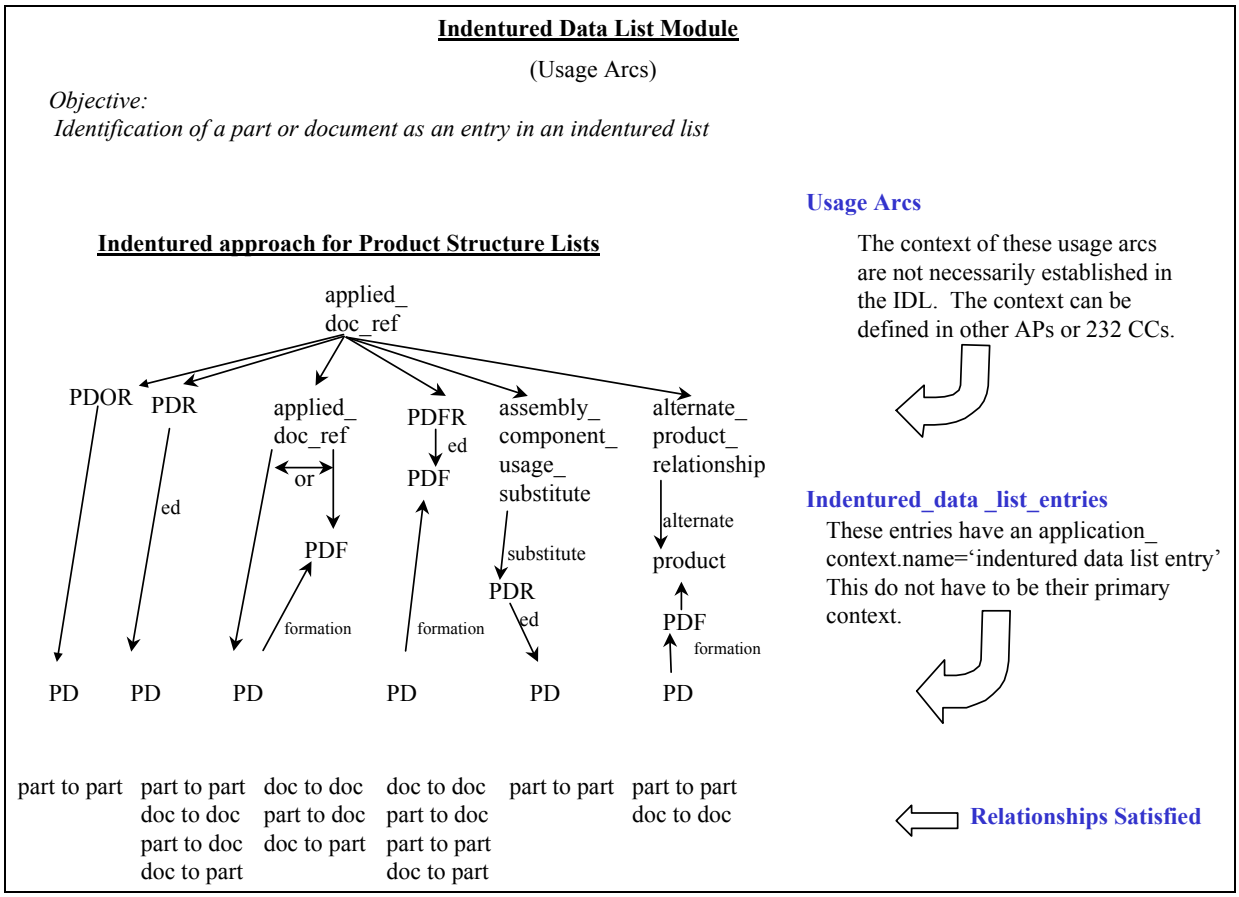


Figure 8.2 - Indentured Data List (Usage Arcs)

The indentured hierarchy of parts and documents uses a **document_product_equivalence** to tie the IDL body **product_definition** to and equivalent **document** construct. An **applied_-document_reference** instance is use to collect all the associations among the different products such as (parts and document). These associations entities are things such as **product_-definition_relationship**s (and all its subtypes), **product_definition_formation_relationship**s (and all its subtypes), as shown in Figure 8.2. These associations can be referred to as the ' Usage Arcs' of the hierarchy structure. The top-level nodes can also be included in this list established by this **applied_document_reference**. Different types of tabulation methods can be identified through the use of the related **object_role.name** to the **applied_document_reference**.

Each of these selected Usage Arcs contains a parent-child relationship. The child of each relationship can be considered to identify the individual entries in the hierarchy tree. The parent of each relationship can be considered to identify the connecting node. Again the very top node(s) in the tree can be included in the **applied_document_reference** select list to aid in processing.

8.1 Indentured Data List Header

The indentured data list header is the configuration, data management, and usage information necessary to manage and control the collection of data being identified with in an indentured data list document. The indentured data list header contains information such as identification information, references other business documents, and approval information.

The indentured data list header is existence dependent on the document header module, section 5.1. While the document header module is generic for any document, the indentured data list header adds identification constraints, as shown in Figure 8.3, and adds the mapping to relate procurement references to the indentured data list document, as shown in Figure 8.4.

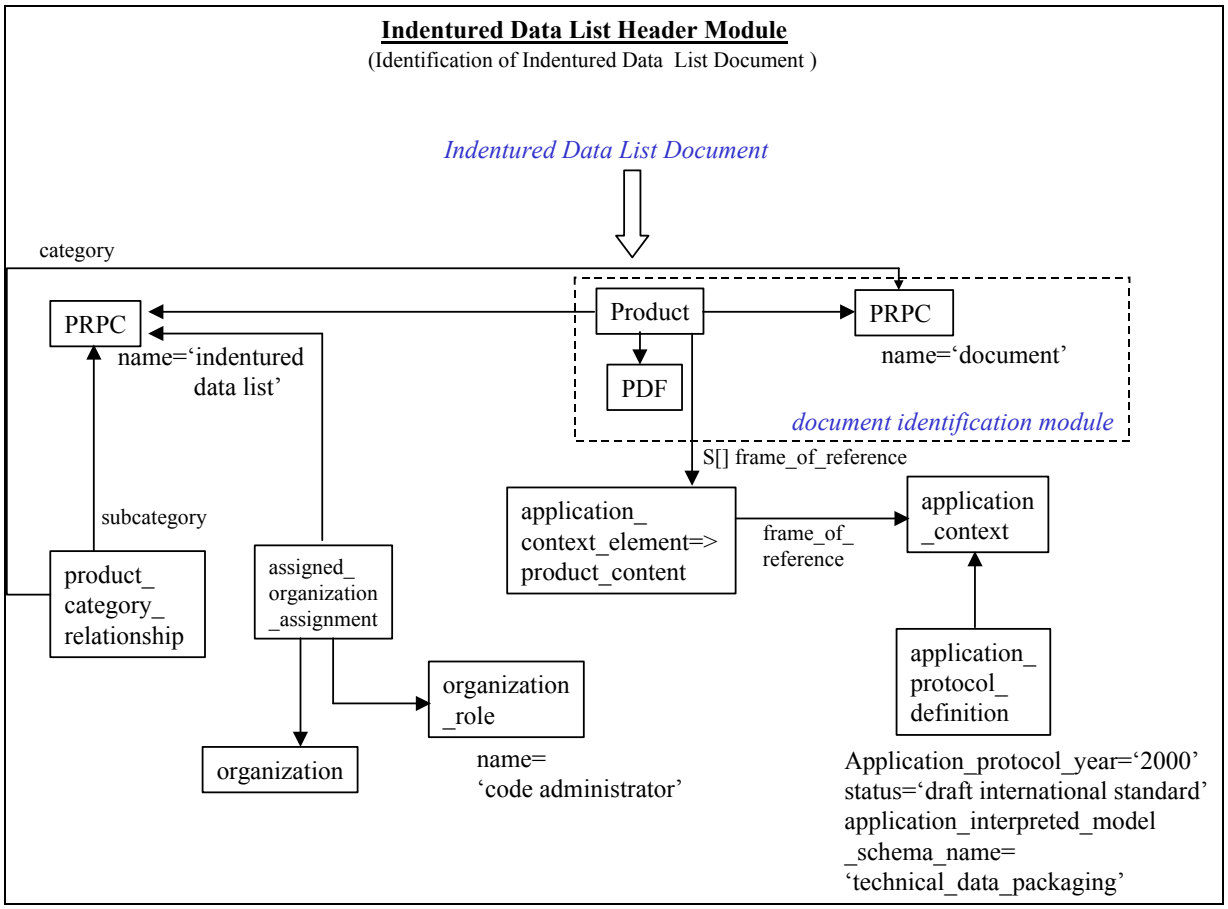


Figure 8.3 - Identification of Indentured Data List Document

8.1.1 Indentured Data List Document Identification

Identifying an indentured data list document is done by relating two **product_reltated_-product_category** (**PRPC**) to a product with their name attributes equal to 'indentured data list' and 'document'. The two **PRPC**s should be related to each other through a **product_category_-relationship** (**PCR**), with the **PRPC.name**='document' instance being associated with the **PCR.category**. An **application_protocol_definition.application_protocol_year** should equal '2000' as shown in Figure 8.3.

8.1.2 Procurement References

The procurement references specifies a part identification or a document identification that was used as justification for development of the Indentured Data List document. The justification is a business purpose that identifies a document or part that was provided by the organization procuring the data contained in the indentured data list document. The procurement references need not be specified for a particular indentured data list header. There may be more than one procurement references for an indentured data list header. Figure 8.4 shows how procurement references are instantiated.

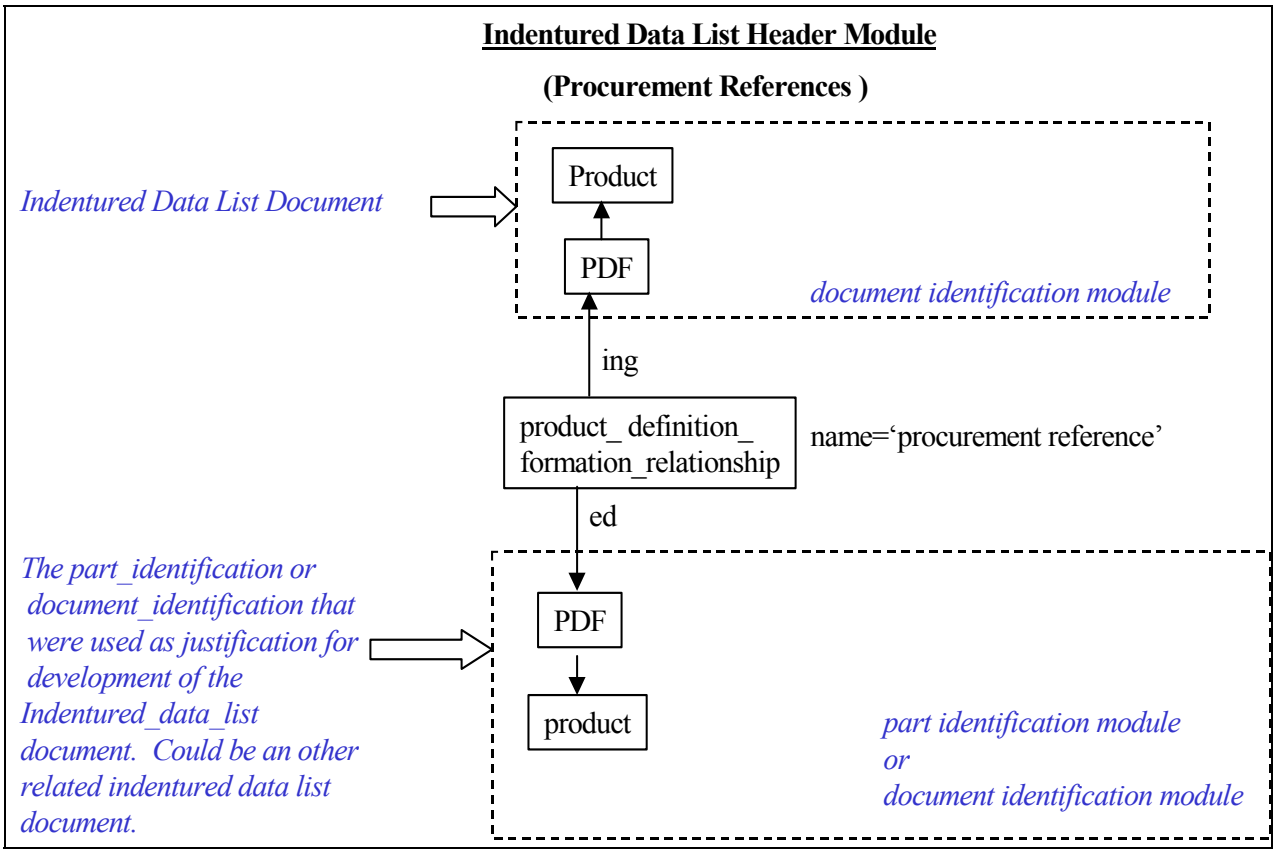


Figure 8.4 - Procurement References for an Indentured Data List Document

8.2 Indentured Data List Body

An indentured data list body contains the main content of the IDL document. The core entity of the indentured data list body is a **product_definition** construct with its description attribute equal to 'indentured data list content.' From this **product_definition** instance the following four sections of the body come together. The body of the IDL document may contain the following:

— list of notes;

— revision history;

— list of standard documents;

— indentured hierarchy of parts and documents.

8.2.1 Indentured Data List Tabulation

8.2.1.1 Indentured Data List Tabulation Method

The indentured data list tabulation method is the same as documented in section 4.4, indentured list method. The only differences are in Figure 6.6 – Indentured Approach for Product Data Structure , Figure 6.7 - Example Indentured by Part for Exchange Management , and Figure 6.8 - Top Nodes Identified in List for Exchange Management

. In an indentured data list there are no references to files (**document_file**) as there are in a tabulation with in an Exchange Management document described in section 4.0. Modified figures of Figure 6.6 – Indentured Approach for Product Data Structure , Figure 6.7 - Example Indentured by Part for Exchange Management , and Figure 6.8 - Top Nodes Identified in List for Exchange Management

reflecting IDL requirements are Figure 8.5, Figure 8.6, Figure 8.7 respectively.

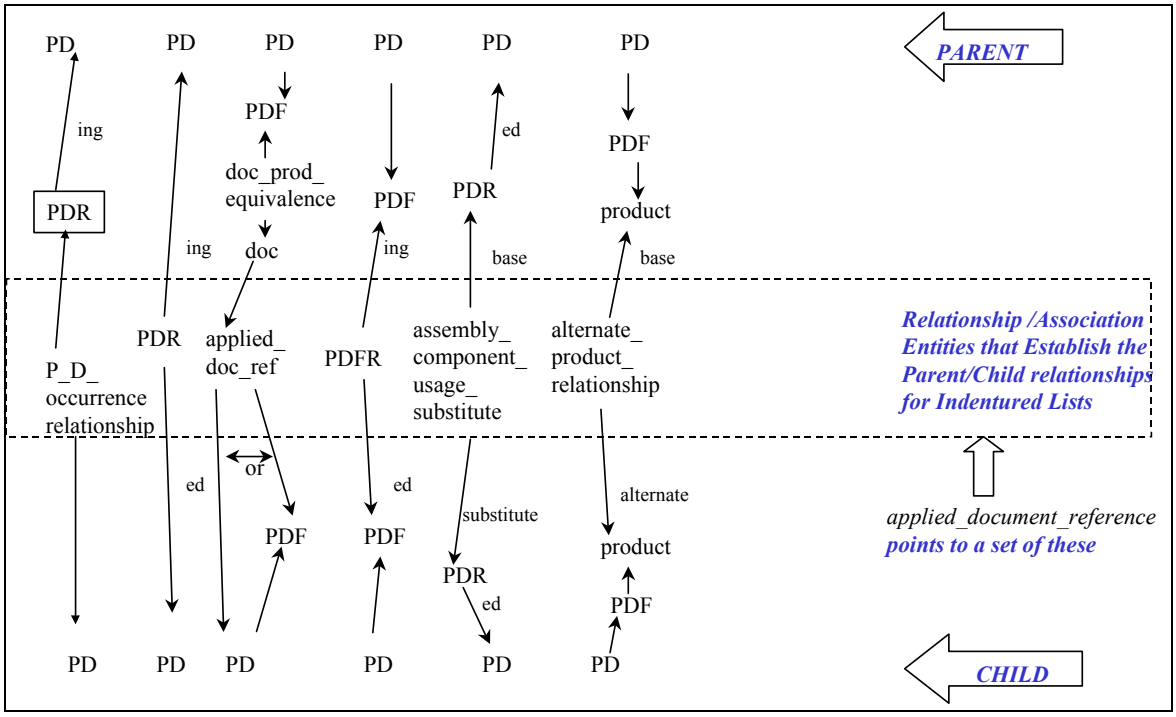


Figure 8.5 - Indentured Approach for Product Structure Lists (IDL)

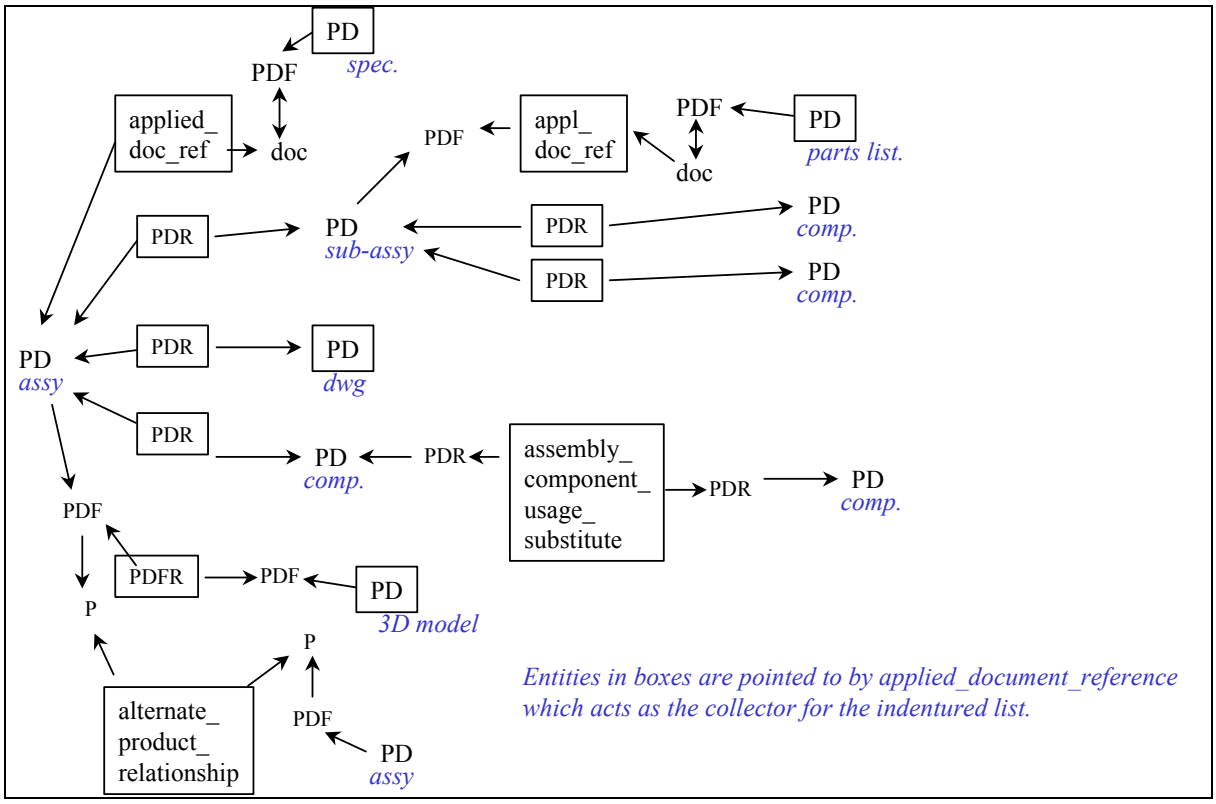


Figure 8.6 - Indentured by Part Example (IDL)

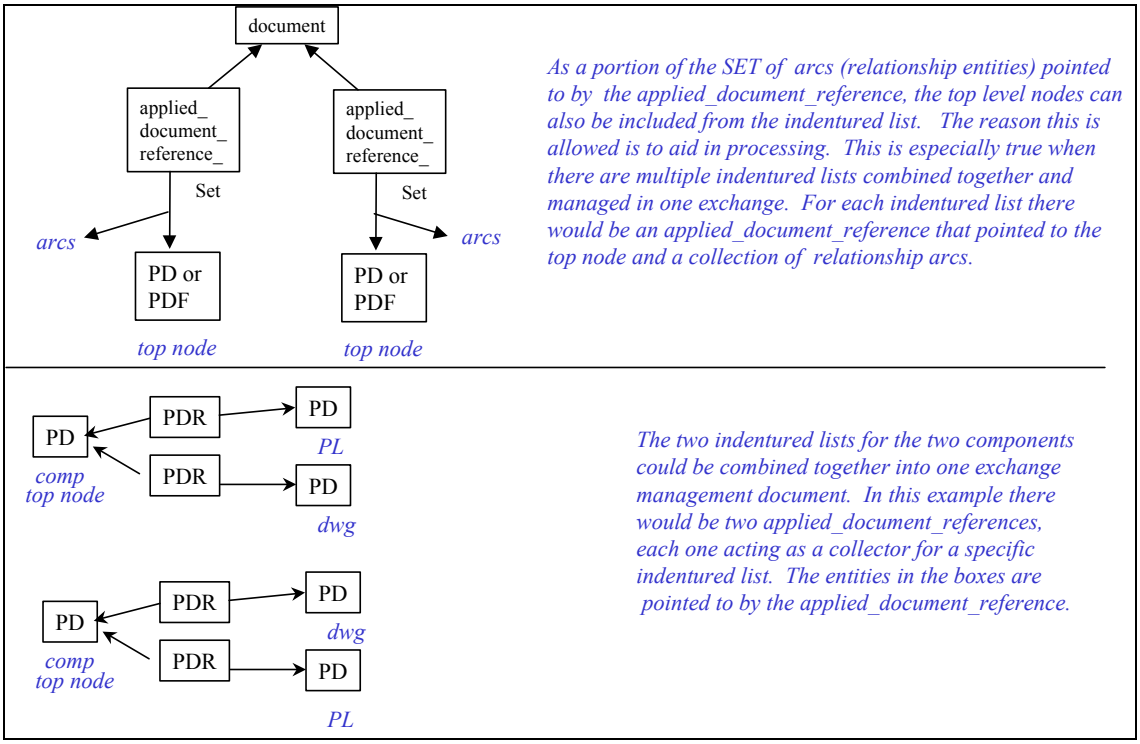


Figure 8.7 - Top Nodes Identified in List (IDL)

8.2.1.2 Indentured Data List Entry

The indentured data list entry is a collection of characteristics about an item (part), or Tdp_element (document) in an Indentured data list document. The collection of characteristics are typical characteristics about the data that is maintained within Product Data Management or Configuration Control systems. The characteristics are employed to manage the use and distribution of the item, or Tdp_element.

8.2.1.2.1 AIM Specification

A **product_definition** represents an indentured data list entry. This **product_definition** can be for a part or a document.

When the **product_definition** is for a document there are multiple possible **application_context**s that could exist. These **application_context**s are described in document_definition module and document_header module. The value for **application_context.name** that is required for an indentured data_list_entry is 'indentured data list entry'.

NOTE a **product_definition** may have more than one **application_context** associated to itself.

When the **product_definition** is for a part, the application_contexts are defined by the part_-definition module.

Figure 8.8 provides the basic paths between the indentured data list entry (**product_definition**) and each of the indentured data list entry's characteristics. Nine characteristic properties are describe that can be applied to the indentured data list entry. Four of the characteristics utilize the same occurrence of **property_definition** in their path. The ten characteristics are the following:

— Indentured Level Tag;

— Special Condition;

— Notation;

— Data Usage Rights;

— Available From – Contract Submission;

— Available From – Document Reference;

— Change Information – Superceded;

— Change Information – Version of the Indentured Data List Document a Entry Applies;

— Effective On.

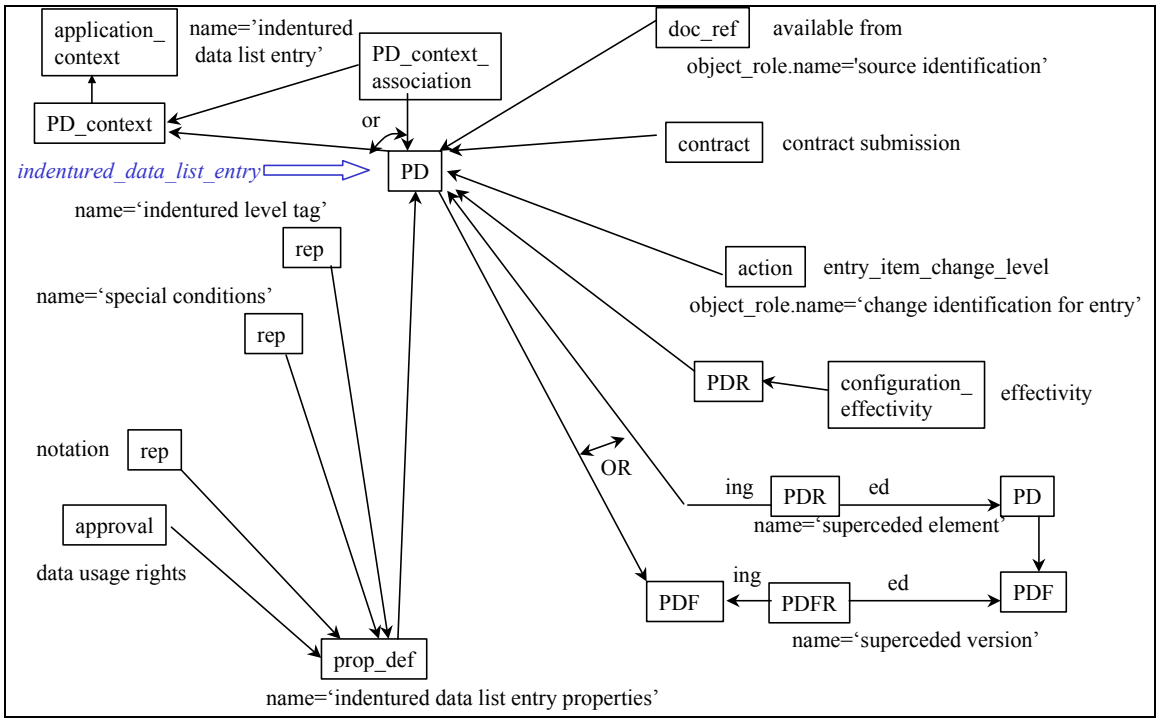


Figure 8.8 - Indentured Data List Entry Characteristics

8.2.1.2.2 Indentured Level Tag

Figure 8.9 shows the indentured level tag entry parameter. The effective_on parameter points to the effectivity module. The indentured_level_notation uses a **descriptive_representation_-item.name** to capture a text notation for a particular indenture level. This can be something like a number (e.g., 1,2, …5) or an alpha character (e.g., A, B, C,…Z, AA). With or without the indentured_level_notation being instantiated, the actual indentured level can be derived from the indentured list. The indentured level tag has optional entry parameters.

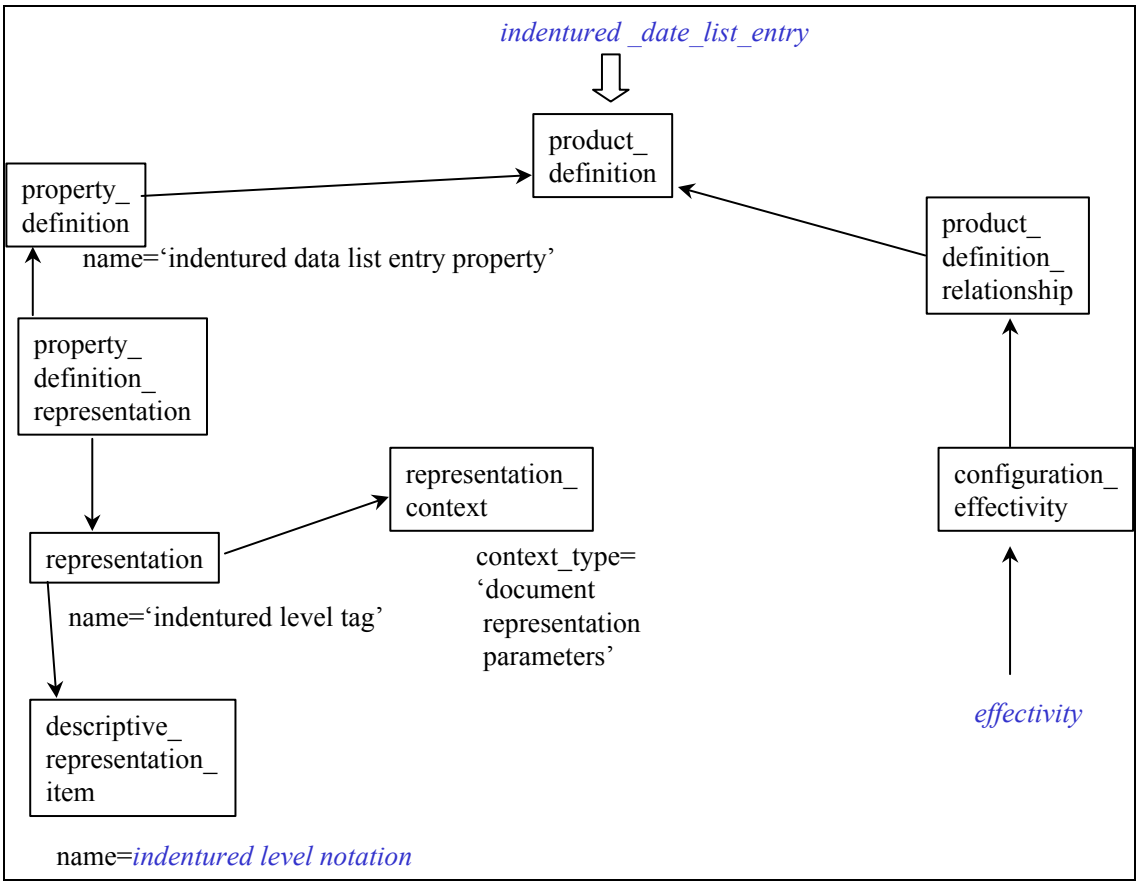


Figure 8.9 - Indentured Level Tag and Effective On

8.2.1.2.3 Special Condition

A special condition placed on an indentured data list entry is shown in Figure 8.10. This special condition is characteristic of an item or Tdp_element that is mutually agreed to between parties for a business purpose. There are three pieces of information that make up this characteristic. The first is a code that specifies a specific identifier from a set of mutually agreed upon special conditions. The string of characters that make up the code is captured in the **descriptive_representation_item.name** attribute. The second is a description of the special condition. The description is captured in the **descriptive_-representation_item.description** attribute. Both the code and description information are optional but one must exist. The third is a type of coding schema. Coding schemas can be established by companies and industries that agree on different codes to depict different conditions. These coding schema agreements are sometimes contained in industry or government standards. An example of a United States standard that contains a special_condition coding schema is ANSI ASME Y14.34M. The coding schema is captured in the **representation_context.context_type** attribute. Constrained string values are shown in Table 2.

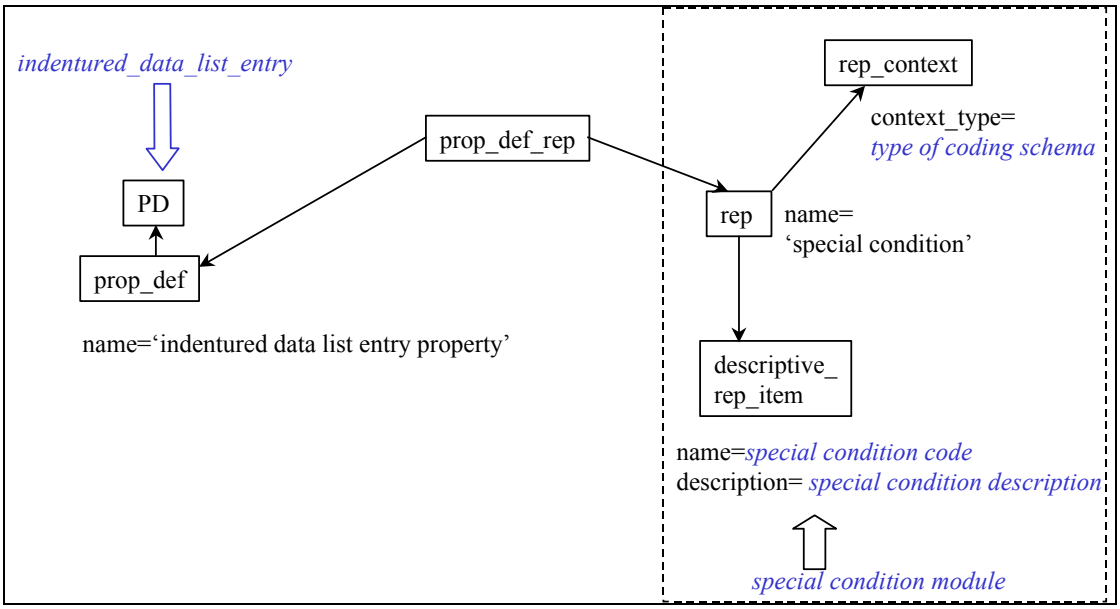


Figure 8.10 - IDL Entry Characteristic (Special Condition)

8.2.1.2.4 Notation

Figure 8.11 shows how to capture general notation about an entry. Notation is a human interpretable string of characters. The general concept of notation is defined in its own module, described in section 6.1. When notation is used as notation for an indentured_data_list_entry, **representation_context.context_type**='document representation parameters'.

8.2.1.2.5 Data Usage Rights

Data usage rights are the rights the destination user has with this data. An instance of the entity approval is used to capture data rights information, see Figure 8.11. The **approval.level** attribute captures the specific text that describes the data usage rights of the user of the data the entry represents. An example could be stipulating the data usage rights to be 'Limited'. The corresponding **product_definition** (Data_definition_entry) is associated with this approval through a **property_definition** and an **applied_approval_assignment**. The corresponding attributes are constrained to uniquely identify the path. Associated with **applied_approval_-assignment** are **property_definition.name**='indentured data list property entry' and **object_-role.role**='data usage rights' The values for **approval_status.name** will follow the PDM Schema RPG for approval module. Data Usage Rights is an optional entry characteristic.

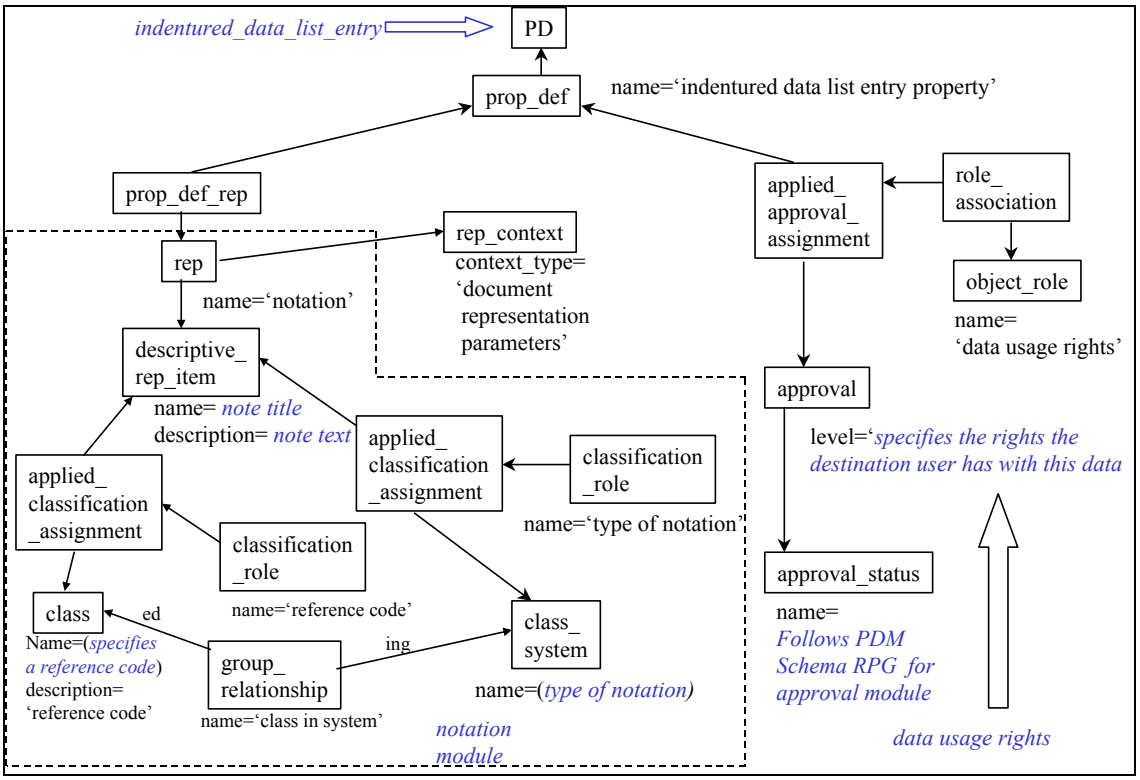


Figure 8.11 - IDL Entry Characteristic (Entry Note, Data Usage Rights)

8.2.1.2.6 Available From – Contract Submission

Available from is an indentured data list entry characteristic that either identifies where the referenced entry data is obtainable from, whether through a contract or some referenced document (Figure 8.12). The contract submission identifies a contract, the location of the contract submission, and date of submission. The contract is applied to the indentured data list entry (**PD**) using an **applied_contract_assignment** with a corresponding **object_role.role** attribute constrained to 'contract submission.' The location of the contract_submision is captured with an organization entity and its organization_address entity. The organization is applied to the **applied_contract_assignment** with the corresponding **organization_role.name**='location of contract submission'. The date of submission is captured by applying the date entity to the **applied_contract_assignment** using an **applied_data_assignment** entity with its corresponding **data_role.name**='date of submission'. When a time is also required to be captured with the date, apply a **date_and_time** entity to the **applied_contract_assignment** entity using an **applied_-date_and_time_assignment** with a **date_time_role.name**='date and time of submission'.

8.2.1.2.7 Available from - Document Reference

A reference document can also identify from where the Indentured_data_list_entry is available, see Figure 8.12. This reference document contains the available from information. The reference document is mapped to a **product** and associated to the Indentured_data_list_entry (**PD**) through an **applied_document_reference** with the associated **object_role.role**='source identification'. The identification of the reference document follows the document_identification module RPG.

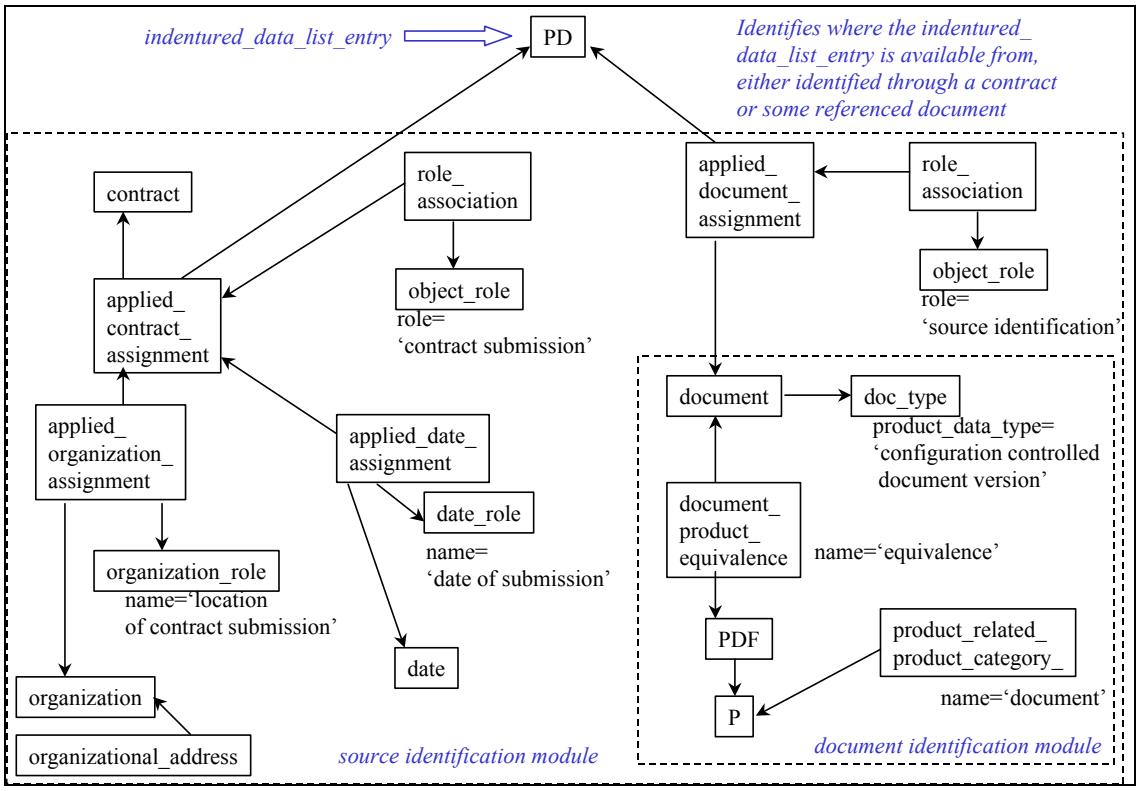


Figure 8.12 - IDL Entry Characteristic (Source Identification)

8.2.1.2.8 Change Information – Superceded

Superceded change information establishes a relationship between a document or part in the current list of entries and the one it replaced from the previous version of an indentured data list. This relationship is captured by instancing a path through a **PDR** relationship or a **PDFR** relationship. The entry that is being replaced or superceded is identified with the 'related' attribute in the relationship entity, **PDR** or **PDFR**. The name attributes of the relationship entities are constrained to **PDR.name**='superceded element' and **PDFR.name**='superceded version'. Figure 8.13 shows these relationships.

8.2.1.2.9 Change Information – Version of the Indentured Data List Document an Entry Applies

Version change information characteristic establishes which version or versions of the indentured data list document applies to a particular entry. This will provide the capability to track and record all change history of the versioning process of an IDL. The version of a particular indentured data list document is not only captured by a **PDF.id** but also an **executed_action** that identifies the change level of a document. The **executed_action** that is associated to the indentured data list entry (**PD**) is the same **executed_action** that was associated to the previous version of the indentured data list document. To capture full history of change activity between two enterprises an **executed_action** for each previous version of the indentured data list document would exist and each would be associated to the appropriate indentured_data_list_-entries. An indentured_data_list_entry could be identified as being valid in more than one version of the indentured data list document. An **applied_action_assignment** with its associated **object_role.role**='change identification for entry' is used to connect the change_identification to

the indentured data list entry (PD). Figure 8.13 shows how this version change information associates with indentured data list entry (PD).

The change identification module, section 5.2, is used in this version change information characteristic. The change identification module will provide guidance for the instancing the appropriate entities such as **executed_action**.

NOTE A **PDF** with its associated executed_**action** having its attribute '**name**='change identification', signifies that this is only change information for this **PDF**(version) of the document. This could be use to facilitate a level of delta change within an exchange scenario.

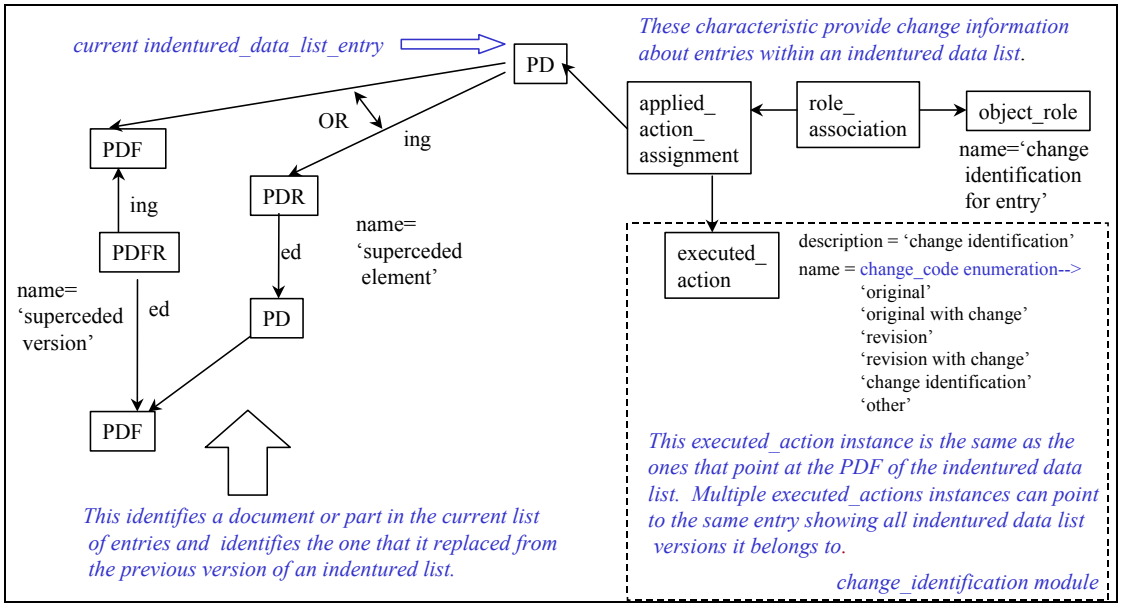


Figure 8.13 - IDL Entry Characteristic (Superceded, Entry Change Level)

8.2.1.2.10 Effective On

Figure 8.9 shows effective on entry parameters. These include two optional entry parameters effective on and indentured level notation. The effective_on parameter points to the effectivity module with its usage document in the PDM Schema usage guide. The effective_on parameter is an optional entry parameter.

8.2.2 Indentured Data List Body Notes List

A tabulation of notes can be collected as a portion of an indentured data list body. As shown on Figure 8.1, this notes list is accumulated through a **property_definition** that has its description attribute value equal to 'indentured data list body property'. The notes are captured in **descriptive_representation_item**(s). Basic notation description is found in section 6.1. and Figure 6.11.

8.2.3 Indentured Data List Body Revision History

A tabulation of Revision History can be collected as a portion of an indentured data list body. As shown on Figure 8.1, Revision History information is accumulated through the **product_-**

definition that has its description attribute value equal to 'indentured data list content'. The revision history is captured in **executed_action**(s). Basic revision history is found in section 6.2 and Figure 5.1.

8.2.4 Indentured Data List Body Standardization Documents List

A tabulation of Standardization Documents can be collected as a portion of an Indentured_data_-list body. As shown on Figure 8.1, a standardized document list is accumulated through an additional **applied_document_reference** pointing to the **document** equivalent of the Indentured_data_list body (**product_definition**). This **applied_document_reference** will have an associated **object_role.name**='standards document'. The **applied_document_reference** will point to a set of one or more **product_definition_formation**(s) representing documents.

8.3 Indentured Data List Rules

The following is the summary of the rules invoked in the mapping table for indentured data list. Please see section 5.5 for rules invoked by the common table.

8.3.1 contract submission requires date and organization

The **contract submission requires date and orgaization** rule specifies that a contract submission must have date of submission and location of the destination organization.

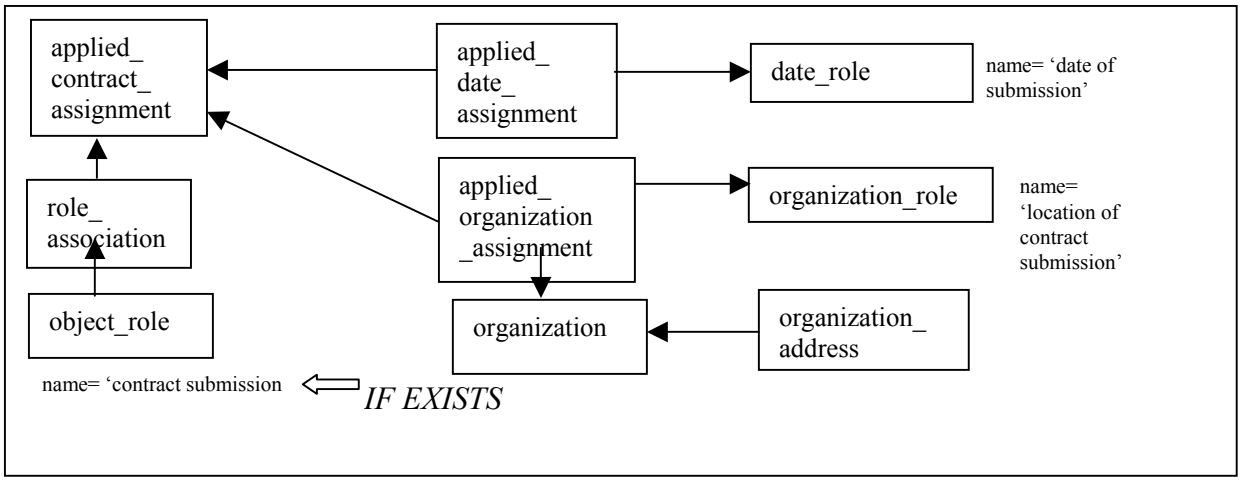


Figure 8.14 - Rule contract submission requires and and organization

8.3.2 Data definition entry string restrict for supersede element

The **data definition entry string strict for supersede element** rule specifies how a superseded relationship is identified through the restriction of a **product_definition_relationship.name** attribute or a **product_definition_formation_relationship.name** attribute. The figure show the case of **product_definition_relationship,** the **product_definition_formation_relationship** case uses the same string restriction.

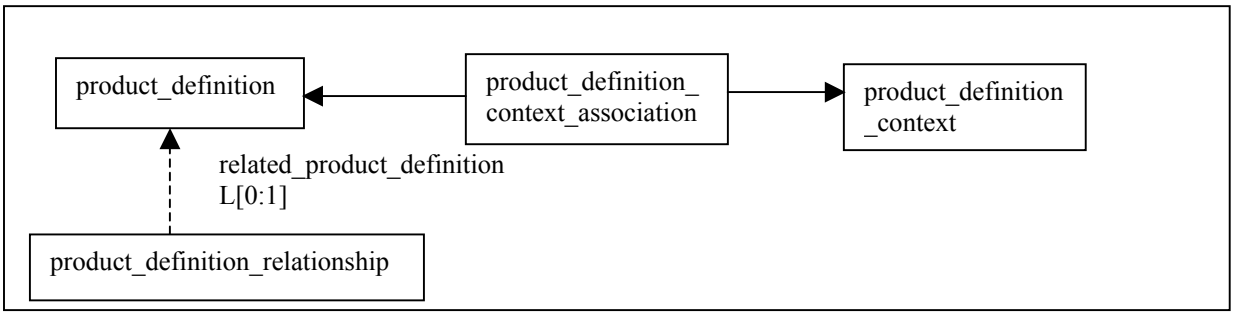


Figure 8.15 Rule data definition entry string restrict for supersede element

8.3.3 indentured data list identification constraint

The **indentured data list identification constraint** rule specifies the string value in **product_category** that identify indentured_data_lists.

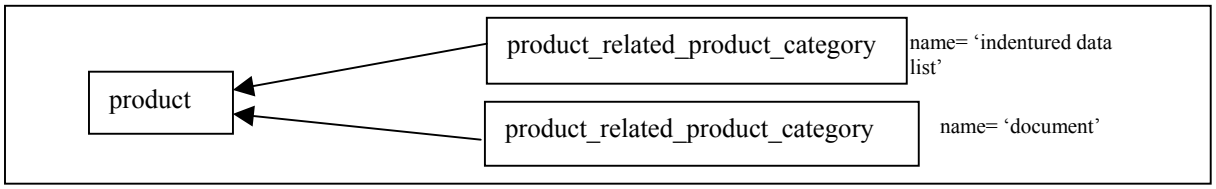


Figure 8.16 – Rule indentured data list identification constraint

8.3.4 indentured level tag identification constraint

The **indentured level tag identification constraint** rule specifies the string value used to identify a human interpretable indentured level for a Data_definition_entry and Indentured_data_list_entry.

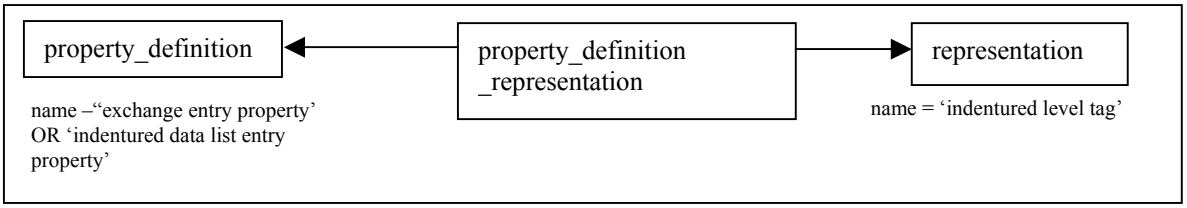


Figure 8.17 - Rule Indentured level tag identification constraint

8.3.5 indentured list method identification constraint

The **indentured list method identification constraint** rule specifies that the three different types of indentured list methods may be applied to only data_definition_exchange and indentured_data_-lists**.**

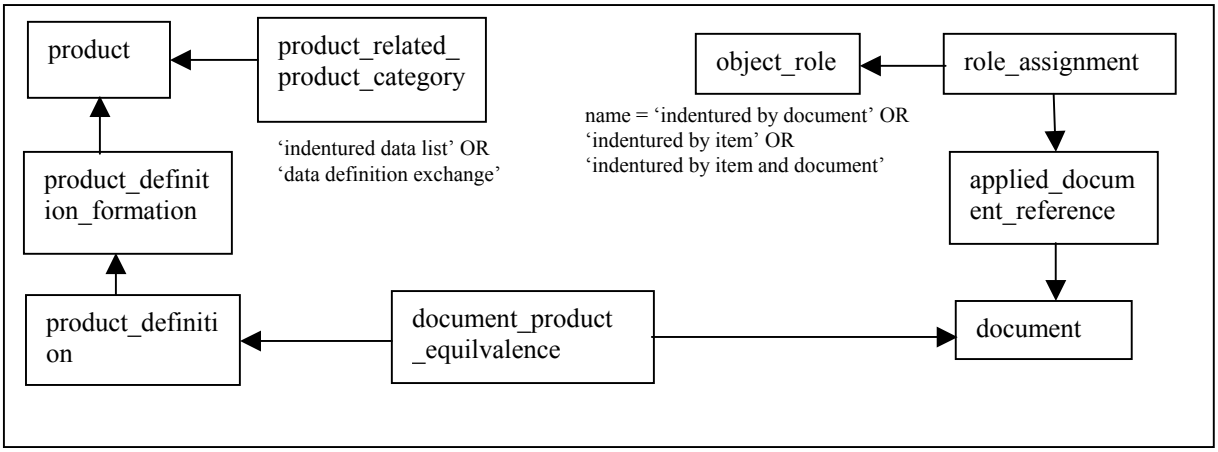


Figure 8.18 Rule Indentured list method identification constraint