DOORS Documentation Method for the Commercial Systems Data Link Organization

Document Number 945-9527-001
Revision C
CAGE Code 0EFD0

Rockwell Collins

Contract Number None

	NAME	TITLE	APPROVAL
Prepared By:	Lori J. Sipper	Preparer	N/A
Approved By:	Hatem I. Abu Dagga	Group Manager	On File
Approved By:	Eileen P. Roberson	DAC Engineer	On File
Approved By:	Mike P. Veit	Technical Project Manager	On File

REVISION HISTORY

VER	REV	DESCRIPTION	DATE	APPROVED
001	-	Initial Release	2012-05-14	R. H. Pulju
001	A	ECO-0422145. FUSN00240426, FUSN00263335, FUSN00265600, FUSN00274668, FUSN00277376, FUSN00280680, FUSN00283307, FUSN00284056, FUSN00300117, FUSN00302374, FUSN00307308	2013-02-04	D. J. Renner
001	B	FUSN00373062 - Grammatical errors , spelling issues, and formatting changes ECO-0632312	2015-07-11	L. J. Sipper
001	C	DLSS-19726, DLSS-9628, DLSS-23837: Updated to support JIRA WP, _Safety Attribute information and for SCL template, formatting changes CO-00177953 REDRAWN	2023-08-03	H. I. Abu Dagga

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Scope

Procedures defined in this document have been developed as a result of lessons learned during Pro Line Fusion initial development. The development methods described in this document should be considered during all development and verification phases of the product life cycle, where DOORS documentation is created or maintained within Rockwell Collins.

1.2 Objective

The objective of this documentation is to provide guidance to the Data Link community within Rockwell Collins Commercial Systems. It identifies the use and maintenance of Specifications, Requirements, Plans, Procedures, Results, and related documentation managed via DOORS Data Link products.

1.3 Acronyms and Abbreviations

CAGEC	Commercial and Government Entity Code
CCB	Change Control Board
CR	Change Request
CT	Change Task
DLSS	Data Link System/Software
DLSS-XXXXX	Abbreviation to indicate a Work Package or Work Element from the JIRA database
DOORS	Dynamic Object Oriented Requirements System
DQE	Design Quality Engineer
DXL	DOORS Extension Language
EAR	Export Administration Regulations
FUSNXXXXX	Abbreviation to indicate a Change Request or Change Task from the change request database
GUI	Graphical User Interface
ID	Identification
ITAR	International Traffic in Arms Regulations
N/A	Not Applicable
OID	Object Identification Number
OLE	Object Linking and Embedding
PDM	Product Development Management
RCPN	Rockwell Collins Part Number
REQ	Requirement
REV	Revision
RWC	Read/Write/Change
SDD	Software Design Document
SQE	Software Quality Engineer
SRD	Systems Requirement Document
SRS	Software Requirements Specification
STP	Software Test Procedure
VER	Version

WE	Work Element
WP	Work Package

2 Background

As requirements, design, and verification tends to flow down and become more exhaustive with document levels in a typical document tree, DOORS (Dynamic Object Oriented Requirements System) provides traceability over different document levels. DOORS is a database of objects that can be designated as requirements objects, design objects, or verification objects. In DOORS each requirement is assigned a specific, unchanging, Object Identification Number (OID). Attribute values are associated with the OID. This facilitates sorting, filtering, and manipulation of objects based on their attributes. From a module, a set of project specific documentation called a view can be created, with traceability among the modules using attributes. Each item in the module is used as an object and is treated as such. This helps create an object level traceability and control.

3 DOORS Documentation

Each product that is part of the Data Link group will have a DOORS administrator to create, control, and manage the project database, whereas a module will be managed by a user. In the Project level, the administrator has full access of read/write/change, and at least one more person has read/write/change access for redundancy. For document/module level, the owner, and any other person(s) he/she designates will have read/write access for a module.

3.1 Document/Module Structure

The technical document structure for a project should be described within the product's planning documentation. The structure is also reflected in the naming convention of the formal modules. (A formal module is a DOORS term and is used to define a module used to capture requirements and/or design, verification results, or proxy information. See DOORS training or help for further explanation.)

Test case modules will be used to document test cases. All requirements will be linked from a test case module whether directly linked, or indirectly linked (ie, a high-level being an ancestor of the test case, likely through the low level requirements). Proxy modules may be used to aid traceability between design, code, tests cases and requirements.

4 Role Descriptions

Project Administrators have DOORS full user privileges. Module managers and Users have DOORS RMCD user privileges.

Upon creation of a formal module, a single DOORS standard user within the project shall be designated as the Module Manager. The Project Administrator shall ensure that all Module managers have Read/Modify/Create/Delete (RMCD) access to the module and that the default module access is set to read-only (R). The module manager and/or Project Administrator may establish appropriate module level access for additional module users or groups.

4.1 Types of Access

Table 1: Types of Access

Access Name	Definition of Access
None	No access to the data, user cannot see the data at all.
Read (R)	Read only access to the data. Assign this access to allow a user to see the data but not to modify the data.
Create (C)	Access to read existing data and create new data items. Assign this access to prevent a user from changing existing data but allow him or her to add new data.
Modify (M)	Access to read and make changes to existing data. Assign this access to allow a user to see and modify existing data, but not add or delete data.
Delete (D)	Access to read and modify existing data and delete data items. Assign this access to allow a user to control existing data, but not allowed to add new data.
Full	Full access to the data, user can read, modify, create, delete, and administer the data.
Admin (A)	Access to change the access privileges of other users. Assign this access to allow a user to have full control over the data. Granting admin access automatically provides read, modify, create, and delete access.

4.2 Project Administrator

The Data Administrator and the DOORS Documentation support team are the project administrators for the project. The project shall have at least two user accounts designated as “administrator”.

A DOORS Project Administrator does the following:

- Start a new project.
- Create and maintain a project.
- Duplicate projects
- Rename projects and modules.
- Generate and maintain list of users for the project.
- Create modules.
- Module importing
- Assist module managers and users.
- Provide System Maintenance in conjunction with IT.
- Archive and restore project and module.
- Delete projects and modules.
- Purge projects and modules
- Unlock modules that are not being edited.
- Maintain standard for documentation.

- Creates Baselines
- Release database to Software Control Library
- Archive Project
- Ensures each formal module has a standard naming convention.
- Ensures that each formal module contains the standardized list of attributes.
- Ensures the Unique ID reference follows a standard convention.

4.2.1 Access Control

The DOORS Project Administrator provides access control through the following:

- Creates a list of groups and adds the users to the groups for a project.
- Create access rights for each user.

The Project Administrator shall ensure that all modules establish and maintain group accesses.

The DOORS Project Administrator may establish access rights to the module for any other project users as needed with Technical Project Manager's approval.

4.3 Module Manager

The manager of a formal module is a DOORS standard user that has been designated as the owner of a module. The module manager has read/write/change (RWC) access privileges at the module level.

4.3.1 Module Access

Upon creation of a formal module, the DOORS Project Administrator and/or the module owner within the project shall be designated as the Module Managers.

4.3.2 Module Status Update

The Module Manager shall be responsible for maintaining the accuracy of the module level attributes, which reflect the maturity and status of the module and its contents.

4.3.3 Module Configuration Control

The Module Manager is responsible for the proper configuration control, change management, viewable image export, and, if necessary, release of module information. Assistance from the DOORS project administrator is available.

4.4 Training and Operational Help

The DOORS Support Team within the Enterprise Tool Integration group at Rockwell Collins has the mandate to train users as required and will provide technical support.

Additional DOORS help can be found within the Rockwell Collins epedia internal website.

5 Module Creation

Modules may be created from an existing template if available. Data can be imported from prior versions of requirements modules.

5.1 Document Front Material

Each of the template modules contains a set of "front material" objects. The front material objects will include the cover sheet with the Title and/or Control Identifier. The objects contain information that provides a common look and feel to any modules that resides in the project repository and is expected to be the master information for document release. Product Family modules will have multiple sets of “front material”, one set for each product version. However, a view is provided for each version such that only “front material” applicable to that particular version is shown.

The front material should contain at a minimum:

- Standard Rockwell Collins cover sheet with Title and Rockwell Collins Part Number (RCPN)
- Appropriate ITAR/EAR notification
- Appropriate CAGEC code
- Revision history, including DOORS baseline for each released revision.
- A note regarding the source DOORS module name and path in the network
- Description of DOORS Attributes that are in addition to the DOORS attributes described in this standard. The minimum information should be:
    - Attribute name (e.g., ~Network)
    - Attribute Type and Values (if enumerated)
    - Attribute usage: How this attribute is to be used by the developers of the module, and how the attribute is to be interpreted by the reader of the module.

5.1.1 Revision History Contents

Each module shall contain a Revision History section. This must be updated with each formal release of the document. It must contain, at a minimum, the following information for each release:

- Revision Identifier - this identifier is the information needed to retrieve the formal release from the Rockwell Collins Software Control Library.
- Date of Release
- Change Drivers – Text or CR/WP Numbers of what drove this change.
- Author – The name of the person that authored the changes to the document for this formal release.

After each release of the document, all CR/WP Numbers should be deleted out of the _CR Number attribute for each DOORS object within the Revision History section.

5.1.2 Version Change Description

Each module shall contain a Version Change Description section. That section is to identify each release of the document that maps to a particular released version of the product that this document pertains to. The content of the Version Change Description section should be as below:

<Change Driver ID> - <Change Driver Summary>

Example:

The Changes made for this revision include the following:

DLNX100000786 - Reworded OID 2726-2728 to include parent text in child requirements

DLNX100001023 - Updated OIDs to have a verification method of Target Test

FUSN00296499 - Preparing for release

DLSS-20158 – Update the SRS to add the ROAAS as an aural warning source

After each release of the document, all CR/WP Numbers should be deleted out of the _CR Number attribute for each DOORS object within the Version Change Description section.

5.2 Module Properties

Each module must have an Object Identifier prefix that is unique for each module within a given project. Typically, these prefix values incorporate the level of requirement (e.g., L1, L2, L3, etc.) as well as a few characters defining the type and name of the document.

Examples include:

L2SUSRD – Level 2 Switch Unit System Requirements Document
L3HMISRS – Level 3 Human Machine Interface Software Requirements Specification
RQHMS – Health Monitor Software Requirements

5.3 Change Management of DOORS Objects

Change management of DOORS objects is not inherently provided by the tool. Therefore, methods described below have been derived to handle changes to documents stored in DOORS.

Changes to requirements will be made using the "_CR Number" and "_Proposed Change" attributes within the DOORS module. Changes include adding new objects, updating existing objects, deleting objects, as well as manipulations to DOORS tables.

5.3.1 Making Changes to Standard Objects

When making changes to standard objects, please use the methods described below.

Please note:

It is important to utilize the syntax described below. Adherence to this syntax will allow for better automation within the change request methods when available.

5.3.1.1 Updating Existing Objects

- When updating the object text field, describe the change within the _Proposed Change attribute using the following format:

  FUSNxxxxxxxx/DLSS-xxxxx:New Text

  Where "New Text" is the proposed updated object text. It is expected that anything after the colon, ":" is intended to overwrite the existing object text.

  In the case where there are multiple changes to the same DOORS object for the same CR/WP, the CT/WE number should follow the CR/WP number, separated by a "-". See 5.3.4 Multiple Changes to the Same Object.

  In the case where the object text includes an OLE object or other non-textual data, the data should be inserted in-line with the proposed change in the same manner as other changes.

- When updating any other attribute of the object, describe the change for each attribute within the _Proposed Change attribute using the following format:

  FUSNxxxxxxxx/DLSS-xxxxx:[*Attribute,New Value*]

  Where "Attribute" is the attribute of the object to be updated, and "New Value" is the proposed new value for the associated attribute.

  In the case where there are multiple changes to the same DOORS object for the same CR/WP, the CT/WE number should follow the CR/WP number, separated by a "-". See 5.3.4 Multiple Changes to the Same Object.

  By "New Value", it is understood that the current attribute will be overwritten with the value of the proposed change. If a user would like to add or delete values from a list, the final list should be reflected in the proposed change rather than just the new or to-be-deleted value.

- Enter the Change Request/Work Package number as FUSNxxxxxxxx/DLSS-xxxxx into the _CR Number attribute.

  In the case where there are multiple changes to the same DOORS object for the same CR/WP, the CT/WE number should follow the CR/WP number, separated by a "-". See 5.3.4 Multiple Changes to the Same Object.

  Do not remove any existing values from this field. This allows the field to be used as a change log, noting all changes to the object since initial baseline.

Note that the above syntax does not make changes to any existing object text or object attributes (other than the _CR Number and _Proposed Change attributes).

Obj ID	For Experimenting with Scripts and Concepts	_Proposed change	_CR Number
Example-1242	**5.6.1 General Requirements**	FUSN00123456:[Object Heading, Overall Requirements]	FUSN00123456
Example-3599	The DLCA **shall** display CPDLC data messages to the flight crew and elicit crew input (HMI).	FUSN00000123:The DLCA **shall** display CPDLC data messages to the flight crew. FUSN00000333:[_Safety,False] FUSN00000333:[~Derived,True] FUSN00000333:[_Assumptions/Rationale,Derived by design.]	FUSN00000123 FUSN00000333

Figure 1: Updating Existing DOORS Object

5.3.1.1.1 Updating Existing Objects That Are Referenced by a Requirement

A DOORS requirement object sometimes references one or more other DOORS objects that are not marked as requirements (e.g., "The page shall operate as specified in the following descriptions."). When a DOORS object that is referenced by a DOORS requirement object is changed, the DOORS requirement object must be indicated as included in the change. This is accomplished by updating the _CR Number attribute of both the changed DOORS object and of the DOORS requirement object that references the changed object with the _CR Number driving the change. This is to ensure that a change to a non-requirement object is properly reviewed under the context of the DOORS requirement object that references the changed object.

5.3.1.2 Deleting Existing Objects

- Describe the change within the _Proposed Change attribute using the following format:
  FUSNxxxxxxxx/DLSS-xxxxx:[DELETE]
- Enter the Change Request/ Work Package number as FUSNxxxxxxxx/ DLSS-xxxxx into the _CR Number attribute.
  Do not remove any existing values from this field. This allows the field to be used as a change log, noting all changes to the object since initial baseline.

When deleting an object, the user should consider the DOORS module hierarchy. When an object is deleted, all objects "below" the proposed deletion will also be deleted. If this is the intended effect, all objects being considered for deletion need to be marked per this method.

Note that the above syntax does not make changes to any existing object text or object attributes (other than the _CR Number and _Proposed Change attributes).

Object Identifier	For Experimenting	_Proposed change	_CR Number
Example-4028	This is an example of a requirement being deleted.	FUSN12345678:[DELETE]	FUSN12345678

Figure 2: Deleting DOORS Object

5.3.1.3 Adding New Objects

During new feature development (where there is large volume of new objects being created), or when an automated method for moving _Proposed Changes to the object's attributes doesn't exist, it can be more efficient use of time to follow the method described below. Basically, the method described here is to indicate in the _Proposed Change attribute that this is a new object, and then make all the changes to the attributes directly. This eliminates the work of staging the attribute values into the _Proposed Change attribute for review, and then moving the attribute values into the attributes after the review is completed.

- Create a new object and populate all the required attributes for the new object directly into the desired attribute.
- In the _Proposed Change attribute, insert:
- FUSNxxxxxxxx/DLSS-xxxxx:[NEW]Enter the Change Request/Work Package number as FUSNxxxxxxxx/DLSS-xxxxx into the _CR Number attribute.
  Do not remove any existing values from this field. This allows the field to be used as a change log, noting all changes to the object since a previous release.
- Populate all other required attributes for the new object directly into the desired attribute.

Obj ID	For Experimenting with Scripts and Concepts	_Proposed change	Rqmt	_CR Number
Example-3749	When and End of Flight event contract occurs (as reported by an FMS event contract), the DLCA **shall** terminate the CPDLC connection and all ADS contracts (for FANS)	FUSN12345678:[NEW]	True	FUSN12345678

Figure 3: Adding a DOORS Object

5.3.2 Making Changes to DOORS Tables

Making changes to DOORS Table objects is necessary as well. Technical data can often be much easier to read and comprehend within a table format. This technical data is subject to change just as any other object. Therefore, the following syntax is in place to make changes to tables.

A DOORS table should always be placed “below” (this is a DOORS hierarchy term) a standard object. The standard object should serve as the caption for the table. This associated object will be hereafter referred to as the “parent object” to the table.

Though requirements standards are specific for each project, it is generally understood that a table or a table cell is not considered a requirement object. The table and each table cell may contain technical data that provides further definition for requirements within the document.

5.3.2.1 Adding a DOORS Table

When adding a DOORS table to the document, the following method is used:

- Create a new object and place “NEW OBJECT” in the new object’s Object Text.
- Describe the change within the _Proposed Change attribute of the parent object using the following format:

  FUSNxxxxxxxx/DLSS-xxxxx:Caption for new table
  FUSNxxxxxxxx/DLSS-xxxxx:[NEW TABLE]

- Enter the Change Request/Work Package number into the _CR Number attribute of the parent object.

  Do not remove any existing values from this field. This allows the field to be used as a change log, noting all changes to the object since initial baseline.

- Insert the table below the parent object.
- Enter the Change Request/Work Package number into the _CR Number attribute of each cell within the table.
- Populate all values within the new table.

Object Identifier	For Experimenting	_Proposed change	_CR Number
Example-4030	NEW OBJECT	FUSN12345678:This is the object text for a new "parent object" for a new DOORS table FUSN12345678:[NEW TABLE]	FUSN12345678

	Column Header
Row Header	This seems like a pretty useless table.

Figure 4: Adding a DOORS Table

5.3.2.2 Deleting a DOORS Table

When deleting an existing DOORS table from the document, the following method is used:

- Describe the change within the _Proposed Change attribute of the parent object using the following format:

  FUSNxxxxxxxx/DLSS-xxxxx:[DELETE TABLE]

Note: The [DELETE TABLE] is used instead of just [DELETE] to ensure that the intended change is to remove the table rather than the parent object

- Enter the Change Request/Work Package number as FUSNxxxxxxxx/DLSS-xxxxx into the _CR Number attribute for the parent object as well as each cell in the DOORS Table.
  Do not remove any existing values from this field. This allows the field to be used as a change log, noting all changes to the object since initial baseline.

ID	For Experimenting with Scripts and Concepts	_Proposed change	_CR Number
Example-4115	Parent Object	FUSN12345678:[DELETE TABLE]	FUSN12345678

	Column Header
Row Header	This seems like a pretty useless table

Figure 5: Deleting a DOORS Table

5.3.2.3 Updating Tables Using _Proposed Change Method

5.3.2.3.1 Adding a Row / Column to a DOORS Table

When adding a row or column to an existing DOORS table, the following method is used:

- Describe the change within the _Proposed Change attribute of the parent object using the following format:

  FUSNxxxxxxxx/DLSS-xxxxx:[NEW TABLE ROW YY]
  or
  FUSNxxxxxxxx/DLSS-xxxxx:[NEW TABLE COLUMN YY]

  Where YY is a descriptor for the new row or column added to the table.

  Note: The descriptor could be the row or column number or could be the row or column "Header". The descriptor should be clear enough to allow the reader to identify the new material for peer review. Rows and columns are numbered starting with "1".

- Enter the Change Request/Work Package number as FUSNxxxxxxxx/DLSS-xxxxx into the _CR Number attribute.
  Do not remove any existing values from this field. This allows the field to be used as a change log, noting all changes to the object since initial baseline.
- Insert the new table row or column as proposed into the table.
- Enter the Change Request/Work Package number into the _CR Number attribute of each new cell within the table.
- Populate all values within the new table row or column.

Object Identifier	For Experimenting	_Proposed change	_CR Number
Example-4046	Parent Object	FUSN12345678:[NEW TABLE ROW Bar]	FUSN12345678

	Column Header
Foo	This seems like a pretty useless table.
Bar	Now the table makes a bit more sense

Example-4057	Parent Object	FUSN12345678:[NEW TABLE COLUMN 3]	FUSN12345678

	Column Header	Column Header
Foo	This seems like a pretty useless table.	3.14159
Bar	Now the table makes a bit more sense	2.71828

Figure 6: Adding a DOORS Table Row/Column

5.3.2.3.2 Deleting a Row / Column from a DOORS Table

When deleting a row or column from an existing DOORS table, the following method is used:

- Describe the change within the _Proposed Change attribute of the parent object using the following format:

  FUSNxxxxxxxx/DLSS-xxxxx:[DELETE TABLE ROW YY]
  or
  FUSNxxxxxxxx/DLSS-xxxxx:[DELETE TABLE COLUMN YY]

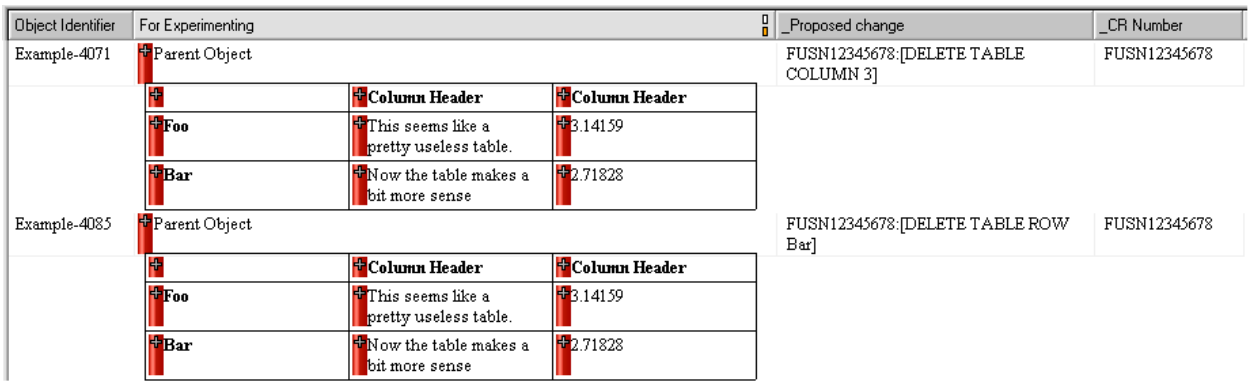
  Where YY is a descriptor for the new row or column added to the table.

  Note: The descriptor could be the row or column number or could be the row or column "Header". The descriptor should be clear enough to allow the reader to identify the new material for peer review. Rows and columns are numbered starting with "1".

- Enter the Change Request/Work Package number as FUSNxxxxxxxx/DLSS-xxxxx into the _CR Number attribute.

  Do not remove any existing values from this field. This allows the field to be used as a change log, noting all changes to the object since initial baseline.

- Enter the Change Request/Work Package number into the _CR Number attribute of each deleted cell within the table.



Object Identifier	For Experimenting			_Proposed change	_CR Number
Example-4071	Parent Object			FUSN12345678:[DELETE TABLE COLUMN 3]	FUSN12345678
		Column Header	Column Header		
	Foo	This seems like a pretty useless table.	3.14159		
	Bar	Now the table makes a bit more sense	2.71828		
Example-4085	Parent Object			FUSN12345678:[DELETE TABLE ROW Bar]	FUSN12345678
		Column Header	Column Header		
	Foo	This seems like a pretty useless table.	3.14159		
	Bar	Now the table makes a bit more sense	2.71828		

Figure 7: Deleting a DOORS Table Row/Column

5.3.2.3.3 Making Changes to a Table Object

When making a change to the text of an existing table object, the following method is used:

Row YY, Column ZZ

- Describe the change within the _Proposed Change attribute using the following format:

  FUSNxxxxxxxx/DLSS-xxxxx:[Row YY Column ZZ,New Value]

  Where "New Text" is the proposed updated object text. It is expected that anything after the colon, ":" is intended to overwrite the existing object text.

  Where "YY" is the descriptor of the row, and "ZZ" is the descriptor of the Column.

  Note: The descriptor could be the row or column number or could be the row or column "Header". The descriptor should be clear enough to allow the reader to identify the new material for peer review. Rows and columns are numbered starting with "1".

- Enter the Change Request/Work Package number as FUSNxxxxxxxx/DLSS-xxxxx into the _CR Number attribute for the parent object and the affected table cell

  Do not remove any existing values from this field. This allows the field to be used as a change log, noting all changes to the object since initial baseline.

ID	For Experimenting with Scripts and Concepts	_CR Number	_Proposed change
Example-4137	Parent Object	FUSN12345678	FUSN12345678:[Row Bar Column 3, 6.02E23]

	Column Header	Column Header
Foo	The constant "pi" is equal to	3.14159
Bar	The molar constant is equal to	6.02000000000000000000

Figure 8: Updating a DOORS Table Cell

5.3.2.4 Updating Tables with In-Line Change Method

Another method for updating tables and their data entails making the changes to the Object Text directly. Changes from one Change Request/Work Package must be fully processed on a particular DOORS object before the changes driven by the next Change Request/Work Package for the DOORS object are applied. This method requires that a tool is available that can compare and showing differences between Adobe© PDF files.

- PDF the version of the DOORS module and save it to a change tracking repository as the baseline version of the DOORS module to be reviewed against
- Make the changes to all the attributes directly in the DOORS object and save the changes.
  - *No changes are staged in the _Proposed Change attribute in this method.*
  - *Note: The DOORS module CANNOT be released at this time. The release process of the DOORS module must ensure that all Change Requests/Work Packages and their associated Peer Reviews are closed.*
- Create a PDF of the updated version of the DOORS module and save it to the change tracking repository as the modified version of the DOORS module to be reviewed for correctness.
- During the Peer Review, use the 'Beyond Compare' tool or another tool that can compare versions of PDF documents to review the differences between the baseline version and the updated version.
- Changes driven by findings in the Peer Review should be incorporated in-line to the DOORS object table.

5.3.3 Developmental Change Requests/Work Packages

A developmental change request/work package is a change request/work package used during initial development of a document prior to the baseline of the artifact.

When a developmental change request/work package is being used, it is acceptable to place the requirement text within the Object Text field and edit the attributes directly. However, it is recommended that the developmental change request/work package number be placed within the "_CR Number" attribute to help provide traceability from that requirement to the baseline peer review.

5.3.4 Multiple Changes to the Same Object

Making multiple changes to the same object is allowed.

Multiple changes to the same attribute in an object with multiple open CRs/WPs are not allowed (i.e. only one open CR/WP against a given attribute can be implemented at any time). The exception to this rule is when CRs/WPs are updating the Document Front Material.

When performing multiple changes to the same object, the description of the new change in the _Proposed Change attribute must be separated from other changes by a new line (a carriage return or line feed separates the objects).

When it is necessary that MORE THAN ONE Change Task/Work Element from the SAME Change Request/Work Package is affecting the SAME artifact at the SAME time, the standard _Proposed Change naming convention (FUSNXXXXXXXX/DLSS-XXXXX) is modified in the following way: *ChangeRequestID-ChangeTaskID (CRID-CTID)/ WorkPackageID-WorkElementID (WPID-WEID)*.

Example: FUSN00001234-FUSN00012345

DLSS-1234-DLSS-12345

This allows development for the current build to be performed with a minimal amount of deviation to the existing process.

If it is planned that both CTs/WEs will be worked at the same time, then both engineers must use the CRID-CTID/WPID-WEID method in their _Proposed Change edits. If the work turns out such that the first CT/WE is not completed before the second CT/WE work must begin, then only the second CT/WE engineer is required to use the CRID-CTID/WPID-WEID.

5.3.5 Concurrent Development

This section describes the process to manage changes when changes only applicable to future builds are being made prior to release of the module in the current build.

5.3.5.1 Modification of Objects for Future Builds

Starting work on module(s) for future release(s) is allowed and does not have to wait for the current changes to be released. This is accomplished by following the proposed changes steps defined in this document and utilizing the following steps:

- When adding a new object:
  - Set _Build ID attribute to the build value that this new object applies to.
- When intending to modify a current object:
  - Make a copy of the current object to a new child object,
    - Note: DOORS allows the links to be copied as well. This will require you to first open each DOORS module where the incoming links start. You will need to open the module in Edit mode (Exclusive or Shared).
    - Delete any links that you do not want in the final approved OID.
    - Add any links that you do want in the final approved OID.
  - Modify _Build ID attribute of the current object with build values it applies to.
  - Modify _Build ID attribute of the new child object with the future build value that applies.
  - Add CR/WP number to "_CR Number" ONLY of the child object. All updates are made only on the new child object, and not the current object.
    - Note: Even though the _Build ID attribute is being modified for the current object, the CR/WP number is not applied to that object. It is understood that when peer reviewing objects for future builds, the reviewer needs to check not only the new child object under review, but also the current object to ensure that its _Build ID was set appropriately. Applying the CR/WP number to the current object as a means to force the reviewer to review the _Build ID attribute would have the undesired effect of marking the current object as being modified for the current build with the future CR/WP number.

Note: The only change made to the current object as part of the update for the future build is setting the Build ID of its attribute to be the current Build ID.

- When intending to delete a current object:
    - Follow the same process used above to modify a current object.

      Note: The new child object will have [DELETE] as its proposed change.
    - When the proposed change gets merged, the new child's object text must be modified to say [DELETE]. This is done so that the deletion request persists until the build containing this change is merged back to the current object, at which point that current object is the object that gets deleted. Scripts that exist that perform merges will need to be updated to follow this desired behavior in this particular concurrent development scenario.

5.3.5.2 Release of Module for Current Build

When the time comes to release the current module, a filter is applied to see objects that either have no _Build ID set (the default), or if a _Build ID is set, matches the one that goes with the current release.

5.3.5.3 Transition of Module to Next Build

After the module has been released and the next build is being worked, all changes that were marked to have been in the next build are then merged into the just-released document, making the next build the new current build.

- For objects that were added that have the _Build ID value that now corresponds to the current build, simply clear the _Build ID value.
- For objects that were either marked to be modified or deleted in what is now the current build:
    - Inspect the history of the child object and apply each of those actions to the parent object in order.
        - Note: Objects that have incoming links will have to have those incoming links adjusted to now point at the parent object. This is accomplished by opening each module from which an incoming link starts (in Exclusive or Shared Edit mode) and correcting the links there.
    - Clear the _Build ID of the parent object (it has now been updated to reflect the changes for the new build).
    - Delete the child object (its changes have effectively been "merged", so it is no longer needed)
        - Note: Ensure that there are no child objects of this object first. That would happen if three or more separate builds were being worked concurrently. In that case, take the child object of the object-to-be-deleted and move it to make it a child of the to-be-deleted child object (now they are siblings). Then the to-be-deleted child object can be deleted.

After this merge, provided that there is no additional concurrent development going, all _Build IDs for all objects should be clear, and normal change-managed development should be used.

5.3.5.4 Future Updates and Peer Reviews

Changes made for future builds under this process can be peer reviewed even while the current build has not yet been peer reviewed. However, because these changes will eventually be merged to the current build once the indicated Build ID becomes the current Build ID, the result of that merge needs to be peer reviewed. Therefore, either 1) a Change Task/Work Element should be added to the CR/WP driving the future work or 2) a new CR/WP should be created, clearly labeled as a Merge CR/WP, and assigned to a future build. This will ensure the merge will take place. This way, the work is assigned, and a peer

review can be associated with the merge to ensure that the changes that were previously peer reviewed are still intact.

5.3.6 Applying Proposed Changes

At the appropriate time within the peer review process, proposed changes are implemented. When implementing the changes, the user must not update the "_CR Number" attribute and must only implement the changes specified by the Change Request/Work Package that was peer reviewed.

During the change process, it is important to understand that deletion of an object with links does not remove the associated links; **outgoing links must be deleted prior to deletion of the object**. Incoming links should be marked for deletion in accordance with the change process.

5.3.7 Making Changes without a Change Request/Work Package

Changes to object text can be made without a Change Request/Work Package (CR/WP) or a peer review finding when they do not change the intent of the object. These changes include grammatical errors, spelling issues, and formatting changes such as bolding, underline and indentation. These changes do not require a peer review unless desired by the person making the update.

5.4 Attributes

The standard groups of attributes are common to all modules (and are prefaced by an underscore); however, a module owner may create more attributes as required. Module specific attributes should be prefaced by character(s) other than a single "_" such as the tilde "~" and should be documented in front material of the DOORS module as described in 5.1 Document Front Material.

5.4.1 Requirement Attribute Use

All standard attributes must be completed as directed in this method for each requirement. Optional fields will be subject to review but are not intended to be used for certification credit. This method takes priority over all other plans or directions for DOORS requirements definition.

The list below contains the minimum standard attributes used. The names are assigned via this document and may not be reused with an alternative definition unless specifically allowed by this document. As directed by the set of plan, methods and procedures these attributes must be filled out as applicable and when required.

Table 2 lists out the common Attributes used and whether they are Required/Optional, their Type Properties and whether a CR/WP is needed for changes.

Table 2: Requirements Attributes

Attribute Name:	Required/Optional:	Type Properties:	CR/WP Needed for Changes:
Object ID	Required	OID	Yes
Object Text / Object Heading	Required	Text	Yes
_Allocated to	Required	Enumeration	Yes
_Build ID	Optional	Enumeration	No
_Comments	Optional	Text	No
_CR Number	Required	Text	No
_Derived	Required[2]	Boolean	Yes
_Product Version	Optional	Enumeration	Yes
_Proposed Change	Required	Text	No

Attribute Name:	Required/Optional:	Type Properties:	CR/WP Needed for Changes:
_Assumptions/Rationale	Required[1]	Text	Yes
_Req?	Required	Boolean	Yes
_Safety	Required[2]	Boolean	Yes
_Validation	Optional	Text	No
_Validation Comments	Optional	Text	No
_Verification	Required[2]	Enumeration	Yes
_Verification Comments	Optional	Text	No

[1] Only required when the "_Derived" attribute is equal to "True"
[2] Only required when _Req? is equal to "True"

Note: Attributes with Enumeration Type properties can be extended by each product team.

Note: Within DOORS, it is possible to note which attribute changes will result in a change to the color of the change bar for each object. It is recommended that any attributes that require a Change Request/Work Package also make use of this feature (except the "_CR Number" and "_Proposed Change" attributes).

Object ID

This object uniquely identifies the requirement and is auto populated by the DOORS tool.

Object Text / Object Heading

This attribute holds the text generally considered the main text for the module. This may hold requirement text, design text, or other types of text generally holding the technical data.

> *An object should either be a text object or a heading object. There should not be any circumstances where the same object holds both object heading and object text data.*

_Allocated to

This attribute identifies the lower level systems, modules, or components that are expected to provide traceability back to the associated requirement.

The values of this enumeration are specific to each requirements module, and enumerated values should be limited to systems, modules, or components at a lower level to the module.

This attribute is only required to contain data when _Req? = TRUE.

The _Allocated To attribute does not apply to Software level modules, and should not be present as a DOORS attribute in the following types of modules:

- Software Requirements (neither High-Level nor Low-Level) modules
- Software Design documents or traceability modules
- Software Test Case/Procedure documents or traceability modules

_Assumptions/Rationale

This attribute is used to document significant supporting rationale why this requirement choice was made. It may also contain the assumptions that were made when this requirement was written (i.e., what must be true for this to be a valid requirement). All derived requirements must identify rationale which may affect safety.

_Build ID

This attribute is used to identify the release that the functionality would be implemented in. The values of the enumeration used are specific to each requirements module.

_Comments

This is a general comments attribute of type text.

_CR Number

This text attribute is used to document the change request/work package (CR/WP) number(s) that corresponds to change(s) of a managed object text or object heading and the description of change per CR/WP. CR/WP Number is used for both developmental and 'normal' CR/WP numbers.

The full CR/WP number should be documented in this attribute. The format of the text placed in the Change Number field should be the entire Change Request/Work Package number, including prefix (e.g., FUSN/DLSS) and ID (a number). CR/WP numbers should be separated by a new line, a space character, or some other delimiter that is consistent for the particular module.

_Derived

This Boolean attribute is set to "True" when the requirement is considered derived and should not link to a higher level document. The attribute is set to "False" when at least one higher level requirement document drives the specified requirement.

_Product Version

This enumeration is specific for each module and can be used to specify variations to the requirements modules for different products.

_Proposed Change

This text attribute is intended to capture any proposed changes to the associated object attributes. As proposed changes are reviewed and approved, the content of this attribute is copied to the object attributes. See section 5.3.6 for the specific usage of this attribute.

Refer to Section 5.3.4 regarding multiple changes to the same object.

_Req?

This Boolean attribute is used to identify if the associated object is a requirement or not. Only text objects stating "shall" should be designated a requirement. Heading objects, front material, and boilerplate text objects are not requirements.

This attribute shall be set to "True" for every object that states a requirement and shall be set to "False" for a non-requirement object.

_Safety

This attribute is used to indicate a requirement is driven by the system safety assessment. This attribute is set by engineering during creation of new requirements or modification of existing requirements. The System Safety Engineer will review and approve the '_Safety' attribute of all new or modified requirements; any issues found by the System Safety Engineer will be corrected through peer review findings or Change Requests / Work Packages.

_Validation

This is an optional text field used to describe the method used to validate the associated requirement.

_Validation Comments

This is an optional text field to document any comments pertaining to the _Validation attribute.

_Verification

Identifies verification methods such as the following: Inspection, Analysis, Demonstration, and Test. All other non-requirements shall be blank within this attribute.

> *Note: The verification methods listed above are not a complete list and are expected to be clarified within program-level documentation.*

_Verification Comments

This is an optional text field to document any comments pertaining to the _Verification attribute.

5.4.2 Attributes For Other Modules

The following table provides attributes that are populated in non-requirements modules (design documents, proxy modules, verification modules, etc). The definitions for these attributes match the definitions described in the section above.

Table 3: Attributes for Other Modules

Attribute Name:	Required/Optional:	Type Properties:	CR/WP Needed for Changes:
Object ID	Required	OID	Yes
Object Text	Required	Text	Yes
_Comments	Optional	Text	No
_CR Number	Required	Text	No
• In STP Modules	Optional	Text	No
_Product Version	Optional	Enumeration	Yes
_Proposed Change	Required	Text	No
• In SDD modules	Optional	Text	No
• In STP modules	Optional	Text	No

5.5 Peer Review

The draft documentation must be reviewed for approval to ensure the accuracy, applicability, consistency, and quality of the documentation. The process for holding a documentation peer review is detailed in the project's peer review process.

During the appropriate time in the peer review process, any required changes that are listed in the "_Proposed Change" attribute with that CR/WP(s) are incorporated into the DOORS module.

5.6 Standard Module Views

Common views are useful to ensure consistency when reproducing released modules or when performing peer reviews within DOORS modules. The following views will be defined within each DOORS module. When a user decides to define additional views for personal use, it is strongly recommended that the view is saved as a "private view".

5.6.1 Common Deliverable View

A Common Deliverable View will include the following columns:

Object ID
Object Text
_Req?
_Derived

_Assumptions/Rationale
_Safety
_Verification
_Allocated to

The Common Deliverable View will be used for document releases and may be used for Peer Review exports as well.

5.6.2 Peer Review View

A peer review view will include elements within the Common Deliverable View as well as the "_Proposed Change" and _CR Number attributes, any project-level required fields, and any other fields pertinent to the peer review.

5.7 Printing

Documentation may be printed from DOORS as seen on a view. Many document views can be created using different selections of text and attributes. A user may create a new view definition, which can be used as the basis for a new printed report. It is strongly encouraged that these views be saved as "private views" within the DOORS tool.

5.8 Documentation Creation

The Common Deliverable View will be used when delivering DOORS documents to a customer or releasing the document to the Software Control Library.

6 Link Modules and Linking

6.1 Creating Link Modules / Sets

For each upper level requirement that allocates down to a lower level document, at least one requirement link will be provided back from the lower level document to the upper level requirement.

Regardless of the verification method selected, a Test Link shall be provided to every document requirement, allocated or derived from its Test Cases and Procedures document. A test link is required even though the selected verification method is "Lower Level".

A Design link shall be provided to every requirement, allocated or derived, at its lowest level of allocation from the Design documents.

6.2 Link Direction Between Modules

All links shall be created using the downstream module as the link source and the upstream module as the link target.

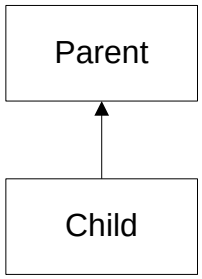


Figure 9: DOORS Requirement Link

[Commentary: It is extremely important that the links be made in the direction indicated. If links exist in both directions between modules in the same link module, a condition may exist that would cause any traceability analysis DXL scripts to enter an infinite loop. It is therefore strongly recommended that the upstream object that is to be the target of a link not be in an editable mode when the link is made. Since DOORS stores the actual link information in the object that is the source of the link, this will prevent a link in the wrong direction from being made. Either open the target module in read-only mode or in edit-share mode and never lock the target object's section.]

The following diagram provides guidance on the structure of linking.

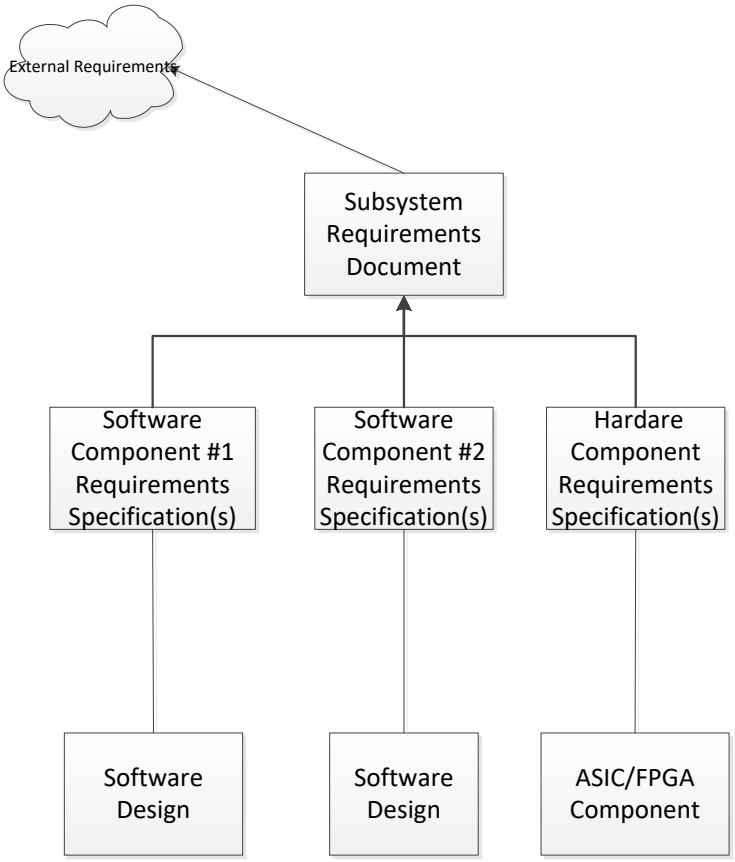


Figure 10: DOORS Documentation Tree

6.2.1 Requirements Allocation and Linking

Each module manager is responsible to ensure that all allocated upper-level requirements have been addressed and linked. As the design progresses the module manager is responsible to ensure the requirements in their modules are allocated as required.

For versions of DOORS greater than DOORS version 9.0, link history is maintained by the DOORS software. Linking of requirements is physically accomplished and tracked via the built-in DOORS links.

For engineers working in close proximity to the DOORS servers, using the DOORS "Link→Start Link" then "Link→Make Link from Start" method or DOORS method of dragging the source DOORS object on top of the target DOORS object is the preferred method.

For work being performed remotely where network performance is poor, linking can also be accomplished by using a link attribute column for each target DOORS Module. When a link is needed to a particular target DOORS module object, the Absolute Number for the target object should be entered in a temporary attribute column of the source DOORS Module. Then the DOORS "Link→Advanced→Link by Attribute…" method should be used to establish the out-links.

6.2.2 Link Traceability

Traceability between two modules is accomplished by using the DOORS physical linking method.

The DOORS software package provides an easy built-in way for individuals to traverse links.

It may be necessary later to update a link. Changes to out-links will be performed using the DOORS physical link method described in section 6.2.1 Requirements Allocation and Linking of this document. All links are subject to the change management process.

6.2.3 Link Validation

Link validation is a procedure in which an engineer responsible for a document ensures that the lower level (child) object or objects linked to the higher level (parent) allocated object requirement meets the following criteria. When all the criteria are satisfied the allocated link combination has been validated.

Requirements Link Validation Criteria

The following criteria must be met for a requirement link to be valid:

- The higher level object to which the lower level object has been linked is a requirement.
- The higher level object to which the lower level object has been linked has allocated the requirement to the lower level project component or module.
- The "_Req?" attribute has been set to "True" on the lower level object which links back to the higher level object.
- The lower level object that has been linked to the higher level object has been allocated via the "_Allocated to" attribute to a lower level module.
- The validator agrees that the linked lower level document link(s) meet the allocated requirement.

The following are criteria for invalid requirement links and the link must be removed:

- The higher level requirement was not allocated to the linked lower level module.
- The higher level object is not a requirement.

Test Link Validation Criteria

The following criteria must be met for a test link to be valid:

- The "_Req?" attribute has been set to "True" on the destination object of the link.
- The higher level object to which the lower level object has been linked has allocated the requirement to the lower level project component or module.
- The validator agrees that the test case document link(s) will verify the linked requirement.

The following are criteria for invalid test links and the link must be removed:

- The higher level requirement was not allocated to the linked lower level module.
- The higher level object is not a requirement.

Design Link Validation Criteria

The following criteria must be met for a design link to be valid:

- The "_Req?" attribute has been set to "True" on the destination object of the link.
- The validator agrees that the design attributed to the design link will meet the linked requirement.

7 Baselines

DOORS provides a versioning capability through the usage of DOORS baselines. Baselines can be used to help identify configuration baselines of requirements or other documents managed through DOORS. Each time a baseline is generated, an additional copy of the module is stored on the DOORS servers. Therefore, baselines should be used to coincide with product deliveries. Projects should strive to baseline a document once every 6-12 weeks.

At the time of baseline generation, the baseline comments should reflect, in an easily readable form, the reason for this baseline and an overview of its content or the changes for this baseline from the previous baseline. It is not necessary to include the list of change request/work package numbers used to modify the document from the previous baseline. In fact, inclusion of the change request/work package numbers in the comments is redundant. The change request/work package numbers are already required to be included in the Revision History, as documented in 5.1.1 Revision History Contents.