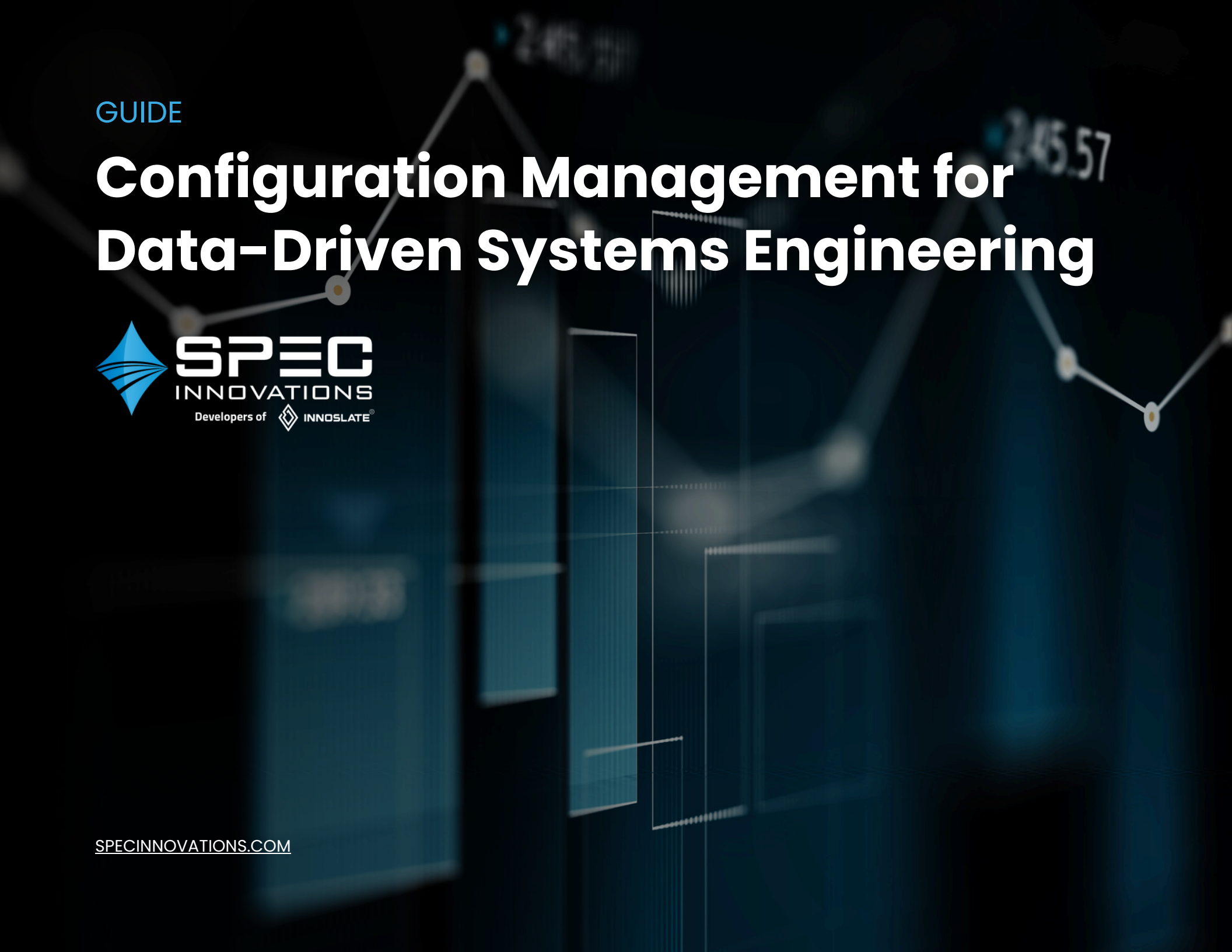
GUIDE

Configuration Management for Data-Driven Systems Engineering



SPEC INNOVATIONS
Developers of INNOSLATE®

SPECINNOVATIONS.COM

INTRODUCTION

Data-Driven Systems Engineering (DDSE) poses distinct challenges in Configuration Management. This guide aims to outline strategies for implementing Configuration Management (CM) in an Innoslate-powered Digital Engineering Environment (DEE). Innoslate, a comprehensive Model-Based Systems Engineering (MBSE) and requirements management solution, offers a rich feature set that establishes an authoritative source of truth in an integrated DEE.

CONFIGURATION MANAGEMENT

Configuration Management (CM) refers to the methods used for maintaining control of a system's requirements, architecture, state, and supporting processes throughout its lifecycle. It is typically described, to varying levels of detail, within a program's Systems Engineering Plan (SEP) and Configuration Management Plan (CMP).

NIST SP 800-53 CM-2[1] defines baseline configurations as "documented, formally reviewed, and agreed-upon specifications" that "serve as a basis for future builds, releases, or changes." Configuration change controls formalize how changes to a baseline configuration are made. NIST SP 800-53 CM-3[2] states that configuration change control includes:

- Determining and documenting the types of changes that are configuration-controlled.
- Reviewing and approving/disapproving changes that are configuration-controlled.
- Documenting configuration-controlled change decisions.
- Implementing approved configuration-controlled changes.
- Monitoring and reviewing configuration-controlled changes.
- Retaining records of configuration-controlled changes.

Two additional aspects of configuration management are:

- Identifying proposed configuration-controlled changes.
- Implementing change control mechanisms.

Figure 1 shows the CM actions discussed in this guide. These actions fall within three major categories: **Configuration Baselining, Change Management,** and **Configuration Control.**

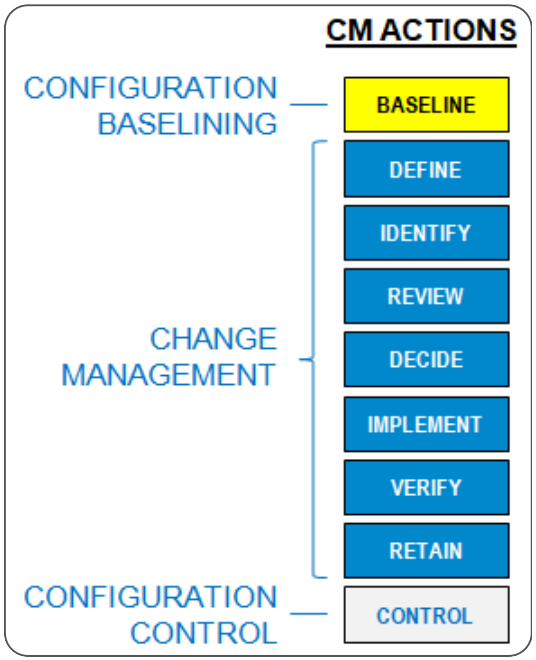


Figure 1. CM Actions

MANAGING CHANGE

CONFIGURATION BASELINING

An Innoslate baseline consists of any approved configuration, such as a specification functional architecture, or physical architecture. It includes specified Innoslate entities along with designated relationships.

BASELINING DOCUMENTS

Innoslate Documents include specifications and plans. They consist of hierarchically sequenced Statements and Requirements entities. Innoslate Documents can be baselined by using Innoslate's Document Baseline feature. This feature leverages entity history to capture and store document baselines (i.e., snapshots in time). Document baselining prevents modifications to the baseline, but not the current state of the entities comprising the document. Document baselining does not prohibit document deletion, which must be addressed through other configuration controls.

Baselined Innoslate Documents should be retained in the organization's CM system. For CM purposes, retain:

- Externally viewable document: Generate from Documents View by selecting 'Report' then 'Basic Document Output (DOCX).'
- Products necessary to reconstruct the document within Innoslate: Generate in Documents View by selecting 'Report,' then:
  - Document ZIP Output (INNO) backup file with the 'Include Baselines' option.
  - Basic Tabular Output (CSV) file with 'Global ID' and 'Relationships' selected.

BASELINING ARCHITECTURES

Architectures include any set of related Innoslate entities. Typically, this includes entities that support physical, functional, or hierarchical models. These entities may align with DoDAF or UAF Viewpoints. Architectures are baselined through management declaration, which should be supported by CM policies, state what constitutes the baseline, and institute baseline configuration controls. For example, a physical architecture baseline may include component-level Asset entities with 'decomposes,' 'connected to,' and 'performs' relationships while specifically excluding 'decomposed by' relationships.

Baselined architectures should be retained in the organization's CM system. For CM purposes, retain:

- Products necessary to reconstruct the document within Innoslate: Generate from within Database View by selecting 'Report,' then:
  - Innoslate ZIP Output (INNO) backup file with the 'Include Baselines' option.
  - Basic Tabular Output (CSV) file with 'Global ID' and 'Relationships' selected.
- Key portions of architecture as an externally viewable document: Generate by navigating to the architecture's root entity, opening in Entity View, selecting 'Report,' and 'Hierarchical Report (DOCX)' with desired diagram types.

CHANGE MANAGEMENT

Change Management is the process of managing changes to configuration baselines. It consists of seven CM actions: **Define, Identify, Review, Decide, Implement, Verify,** and **Retain.**

DEFINE

Policies should be established that define the types of changes that are configuration-controlled at a given point in time. A notional configuration-controlled change policy is shown in Table 1 on the following page. Configuration-controlled changes require formal approval before implementation.

Timeframe	Item	Configuration-Controlled Changes
Post-PDR	SRD Requirement	Name, Number, Description, Rationale; 'refines' Relationship
	Functional Model Actions	Name, Number, Description
	System/Subsystem Assets	Name, Number, Description
Post-CDR	SRD Requirement	Name, Number, Description, Rationale; 'refines,' 'refined by,' 'traced to' Relationships
	SSS Requirement	Name, Number, Description, Rationale; 'refines' Relationship
	Functional Model Actions	Name, Number, Description; 'generates,' 'performed by,' 'receives,' 'traced from' Relationships
	System/Subsystem Assets	Name, Number, Description; 'connected to,' 'performs' Relationships

Table 1. Notional Configuration-Controlled Change Policy

IDENTIFY

Proposed changes to configuration-controlled items within Innoslate should be identified in a Change Request (CR) or Engineering Change Proposal (ECP). CRs establish the impact of proposed baseline changes and document the Innoslate implementation steps. CR formats may vary, but should include the following key elements:

- **Information:** provides administrative information on the CR, including number, title, submission date, originator, urgency (e.g., Routine, Emergency), and type (e.g., Editorial, Technical).
- **Description:** provides a high-level description of what the change will accomplish.
- **Status:** describes the location within the CR workflow.
- **Rationale:** describes why it is necessary to implement the change.
- **Proposed Solution Table:** provides detailed steps for implementing the CR in Innoslate.

The Proposed Solution Table consists of sequentially executed steps to be followed in Innoslate by the implementer. It should minimally include the following columns:

- **Entity Information:** Contains the entity's number and/or name.
- **Location:** Contains the entity's URL.
- **Change Description:** Contains a description of the change to the entity.
- **Change Category:** Specifies the entity attribute name, label, or relationship to be changed (e.g., Name, Description, Label, Relationship).
- **Change Type:** Specifies the type of change, to include: Add, Modify, or Delete.
- **Current Value:** Contains the current value associated with the Change Category, if applicable.
- **Proposed Value:** Contains the proposed value associated with the Change Category.
- **Notes:** Contains any additional explanatory notes to clarify the proposed change.

REVIEW

CRs are reviewed by a Change Control Board (CCB). The CCB should be comprised of management, CM personnel, and technical experts. CCB meetings held to review CRs should include both the CR originators and implementers in attendance.

DECIDE

The CCB will approve or disapprove proposed baseline changes detailed in CRs. CCB decisions must be formally documented and retained. This can occur within the CR form or through the use of related Decision entities (i.e., Decision 'enabled by' Artifact).

IMPLEMENT

Approved CRs are implemented by a designated implementer with the appropriate Innoslate project permissions. The CR's Proposed Solution Table steps should be implemented in the sequence provided. Failure to adhere to the CR's step execution sequence may result in broken dependencies (e.g., a link to an entity that has not yet been created) or confusion as to which parts of the table remain to be implemented. For large tables, the implementer should consider marking the individual steps as complete.

VERIFY

CM personnel should monitor and verify changes to configuration-controlled data. Erroneous or unauthorized changes should be backed out and logged. For Innoslate Documents, CM personnel should review Post-Baseline Change Reports (PBCRs). PBCRs provide a detailed log of all changes occurring to a document's entities following each baseline. PBCRs should be compared against the approved CR's Proposed Solution Table. For architectures, CM personnel should review CR project modifications by comparing individual entities against the approved CR's Proposed Solution Table.

RETAIN

Retain records of all configuration-controlled changes. If CRs are implemented as Innoslate Documents, retain them for CM purposes as externally viewable documents. To generate an externally viewable document from within Documents View, select 'Report' then 'Basic Document Output (DOCX).'

CONFIGURATION CONTROL

Innoslate employs multiple data control mechanisms that can be used to prevent unauthorized data modifications, as shown in Figure 2.

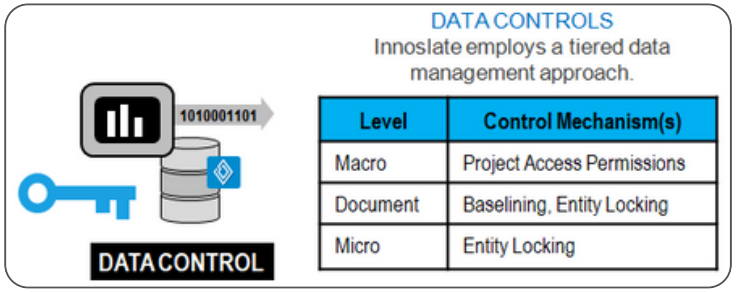


Figure 2. Data Controls

PROJECT ACCESS PERMISSIONS

Project access permissions control the privileges that users have within a project. Segregation of data across projects allows different users and user groups to have permissions appropriate to the program phase. For example, as shown in Table 2, all developer users may initially have Collaborator access to a system-level project. Following PDR, non-SE users may be downgraded to Reviewer access. Optimal use of project access permissions requires upfront planning in conjunction with the development of a Conceptual Data Model (CDM).

Timeframe	Project	User [Permission]
Initial	System	Developer SE Team [Collaborator] Developer Non-SE User [Collaborator] Customer [Reviewer]
Post PDR	System	Developer SE Team [Collaborator] Developer Non-SE User [Reviewer] Customer [Reviewer]

Table 2. Notional Project Access Permissions

DOCUMENT BASELINING

Document baselining prevents modifications to the baseline, but not the current state of the entities comprising the document. Document baselining does not prohibit document deletion, which must be addressed through other configuration controls.

ENTITY LOCKING

Entity locking prevents changes to entity attributes, labels, and relationships. Entities can only be unlocked by the locking collaborator or project owners. Entity locking is frequently used to provide protection from inadvertent or unauthorized changes being made by project Collaborators.

IMPLEMENTING CHANGE MANAGEMENT

CHANGE REQUESTS (CRS)

There are two primary options for implementing CRs within Innoslate. Option 1 consists of an Innoslate-based CR form implemented within Documents View. This CR form can contain either a textual (Option 1a) or dynamically constructed (Option 1b) Proposed Solution Table. Option 2 consists of storing Excel CR forms as Artifact entity attachments. The benefits and required effort for the CR implementation options are shown in Figure 3.

	CR Implementation			
	Option 1a	Option 1b	Option 2	
Benefits	X	X	X	Captures CR as Relatable Innoslate Entity
	X	X	X	Facilitates Innoslate CR Workflow
	X	X		CR Baselining
	X	X		No Reliance on External Tools
		X		Captures Solution Steps as Relatable Innoslate Entities
		X		Dynamically Constructed Proposed Solution Table
		X		Proposed Solution Table Exportable as Spreadsheet
Effort			X	Develop Excel CR Form
	X	X		Develop CR Document Template
		X		Modify CR Document Entity Table Query
		X		Update Schema for Solution Steps
		X		Establish Solution Step Database Query
	Low	Medium	Very Low	End User Training

Figure 3. CR Implementation Option Benefits and Effort

OPTION 1A: INNOSLATE-BASED CR WITH TEXTUAL PROPOSED SOLUTION TABLE

CR Implementation Option 1a consists of creating and saving a CR template within Documents View. First, use the Schema Editor to create a "CR Document" label for class Artifact. Then, create a blank document of type 'CR Document' and add content sections as specified in the Identify section. The Proposed Solution Table should be inserted with the 'Table…' insert option. Back up your document by running a 'Document ZIP Export (INNO)' report.

Add the CR template to the project from Documents View by selecting 'More', selecting, 'Template,' entering, "Change Request," for the template name, and selecting, 'Create.' This will result in a new Artifact named 'Change Request' being created. It is recommended to lock that entity to prevent inadvertent deletion. New CRs are created by selecting document type 'CR Document', template 'Custom Template', and custom template 'Change Request'.

OPTION 1B: INNOSLATE-BASED CR WITH DYNAMICALLY CONSTRUCTED PROPOSED SOLUTION TABLE

CR Implementation Option 1b is implemented in the same way as Option 1a, with the exception of the Proposed Solution Table. Option 1b's Proposed Solution Table is dynamically created by querying the project database for the associated solution steps.

Use the Schema Editor to create a "CR Solution Step" class. "CR Solution Step" should be a subclass of Statement and minimally contain the attributes shown in Table 3.

Attribute	Type	Description
Entity Information	TEXT	The entity's number and/or name
Location	URI	The entity's URL
Change Description	BIG_TEXT	A description of the change to the entity
Change Type	ENUMERATION	The type of change: Add, Modify, or Delete
Change Category	TEXT	The entity attribute name, label, or relationship to be changed
Current Value	BIG_TEXT	The current value associated with the Change Category, if applicable
Proposed Value	BIG_TEXT	The proposed value associated with the Change Category
Notes	BIG_TEXT	Additional explanatory notes to clarify the proposed change

Table 3. CR Solution Step Attributes

Create new CR Solution Step entities from the database view. CR Solution Steps should be numbered sequentially using their corresponding CR number prefix. Add the Proposed Solution Table to the CR template by inserting it with the 'Table Notation…' insert option. Within the 'Insert Table Notation' pop-up, as shown in Figure 4, enter the solution step query, select the 'CR Solution Step' attributes, and enter a query limit high enough to retrieve all of the CR's solution steps.

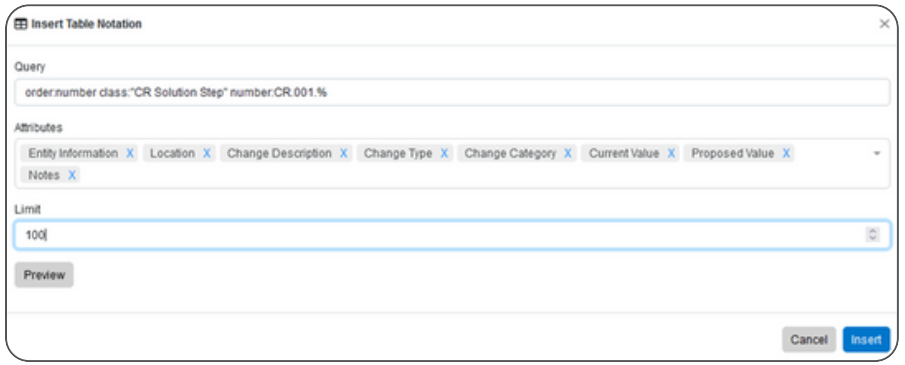


Figure 4. CR Solution Step Insert Table Notation Pop-Up

When new CRs are created from the CR template, the table notation query will need to be updated by the user to retrieve CR Solution Steps corresponding to the new CR number. A CR's Proposed Solution Table can be exported as an Excel spreadsheet for the implementer. From Database View, query and sort for the CR's solution steps (e.g., order:number class:"CR Solution Step" number:CR.###.% where ### is the CR number). Generate an Entity Table Report with the CR Solution Step attributes from Table 3.

OPTION 2: ATTACH EXCEL CR TO ARTIFACT

CR Implementation Option 2 consists of creating an Artifact entity for each Excel CR. This Artifact should be numbered to match the CR number contained within the Excel CR form. The Excel CR is added to the Artifact as an attachment.

CR WORKFLOW

CR workflow can be modeled in Innoslate as a State Machine Diagram] (SMD), shown in Figure 5. The CR workflow SMD can be implemented in a project's schema by:

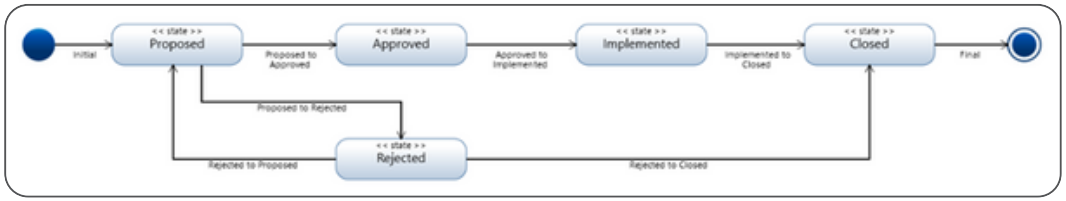


Figure 5. CR Workflow State Machine Diagram

- Creating a 'Change Request' class with an enumerated status field representing CR states.
- Creating a 'Change Request Status' workflow to implement CR state transitions.

IMPLEMENTING CR WORKFLOW STATES

Use the Schema Editor to create a "Change Request" class. "Change Request" should be a subclass of Artifact and minimally contain an enumerated attribute named "Status." "Status" should have values matching the states shown in Figure 5: 'Proposed,' 'Approved,' 'Rejected,' 'Implemented,' and 'Closed.' Set the default value for "Status" to 'Proposed.' If CRs are implemented as Innoslate Documents (Options 1a & 1b), then:

- Use the Schema Editor to update the 'CR Document' label to add the class 'Change Request.'
- Create a CR template based on the 'Change Request' class.
  - If a CR template does not exist, then:
    - Create the document as described in the Option 1a section.
    - Before adding it as a template, navigate to Database or Entity View and transform the source Artifact to the 'Change Request' class.
    - Reopen the document in Documents View and add it as a template.
  - If a CR template already exists, then:
    - Create a new unpopulated document from the CR template.
    - Navigate to Database or Entity View and transform the source Artifact for this new document to the 'Change Request' class.
    - Reopen the new document in Documents View and add it as a template.
    - Lock the new template's Change Request entity in Database or Entity View.
    - Delete the old templates Artifact entity.
  - If desired, navigate to Database or Entity View and transform the source Artifacts for any legacy CRs to the 'Change Request' class.

If CRs are implemented as Artifacts with Excel CR attachments (Option 2), simply create new 'Change Request' class entities for all future CRs. Existing CRs can be transformed from Artifact to the 'Change Request' class.

IMPLEMENTING CR WORKFLOW STATE TRANSITIONS

To implement the CR workflow state transitions, select 'Workflow' from within the Schema Editor. Select 'Add Workflow Class' and the 'Change Request' class. If necessary, select the 'Status' enumerated property. Define the workflow transitions for each 'Status' state by selecting 'Add Transition', the name of the target transition state, users/teams with permission to transition the state, and users/teams notified of the state transition.

For example, in Figure 5, the 'Proposed' state has two transitions. When defining the workflow, 'Add Transition' will be used to select 'Approved,' creating the 'Proposed to Approved' transition, and to select 'Rejected,' creating the 'Proposed to Rejected' transition, as shown in Figure 6.

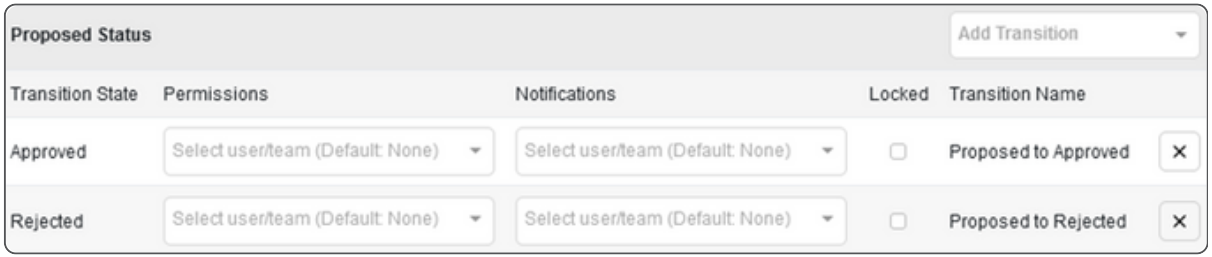


Figure 6. 'Proposed' Status Transitions

The Innoslate CR workflow should be used in conjunction with CR procedures defined in the organization's CMP. From within Database View, CR status reports can be generated using the Entity Table Report with 'Status' selected as a column attribute.

CONCLUSION

Innoslate allows an organization to maintain control of a system's requirements, architecture, state, and supporting processes throughout its lifecycle. It provides versatile features for controlling and tracking change at different levels of granularity. Innoslate provides the capability to implement and supplement processes within an organization's CMP.

REFERENCES

[1] U.S. Department of Commerce, NIST SP 800-53 CM-2, Rev. 5 Security and Privacy Controls for Information Systems and Organizations § (2020).

[2] U.S. Department of Commerce, NIST SP 800-53 CM-3, Rev. 5 Security and Privacy Controls for Information Systems and Organizations § (2020).