|  |
| --- |
| ***AuroraView*** |
| SCD Modernization |
| Detail Design |
|  |
| **AuroraView Team** |
| **10/4/2013** |



Contents

1. Introduction 2

1.1. Related Documentation 2

1.2. Terms and Definitions 2

1.3. Revision History 2

2. Solution Architecture 4

2.1. Overview 4

2.2. Architecture and Technology 4

2.2.1. Physical Architecture (Tiers) 4

2.2.2. Logical Architecture (Layers) 5

2.2.3. Technologies Used 6

3. Solution Components 7

3.1. User Interface Winframe 7

3.2. Project Explorer 8

3.3. Project Management 8

3.3.1. Hierarchy Tree – Menus, screens and fields 9

3.3.2. Folder Management 17

3.3.3. Version Management 34

3.4. Activate Content and Copy content files 38

3.5. Assign/Remove Certificate 39

3.6. Search 40

4. Data Model 42

4.1. Project Explorer 42

4.1.1. ERD 42

4.1.2. Table Definition 42

4.2. Security 48

4.2.1. ERD 48

1.1.2 Tables definition 48

4.3. UserProfile 48

1.2 Messages and Error Handling 49

4.3.1. ERD 49

4.3.2. Table Definition 49

5. APIs and Interfaces to Other Modules 50

5.1. Content Management 50

5.1.1. getTreeObjects function 50

5.1.2. getContentByContentVersionId function 50

5.1.3. Resource Management 50

5.1.4. Project Management 51

6. Messages 52

7. Security 53

8. Assumptions and Open Issues 55

8.1. Assumptions 55

8.2. Open Issues 55

# Introduction

AuroraView LTD is pleased to present SCD with our Detailed Design for AuroraView’s parts of the Modernization project.

This Detailed Design covers mainly the system architecture, the Project Explorer and Project Management functions, the data model and the interfaces.

Our proposed solution is based on state of the art development methodologies and architecture, leveraging Microsoft technologies.

## Related Documentation

|  |  |
| --- | --- |
| Origin | Title |
| SCD | GenPR Modernization Presentation – August 2013 |
|  |  |

## Terms and Definitions

|  |  |
| --- | --- |
| Term | Definition |
|  | TBD |
|  |  |

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | | Revision | Description |
| 30 Sep 2013 | | 1.0 | Initial version |
| 6 Oct 2013 | | 1.6 | Updated Related Project Management section |
|  | |  | Updated Data Model section – added Security, Messages and Error Handling sections, updated Project Explorer Table Definition section |
|  | |  | Added Security section |
|  | |  | Added Messages section |
|  | |  | Added security validations to each section  Added error handling validations to each section |
|  | |  | Updated Project Management section – general assumptions  Added warning and information messages |
| 8 Oct 2013 | 1.7 | Removed Emergency versions management and all related validations |
| 23 Oct 2013 | 2.1 | Updated Security Section  Updated Error Handling  Updated Project Explorer Tables Definition section  Updated Clone Regular project section  Updated Version Management section |
| 24 Oct 2013 | 2.3 | Removed Project Files tab from screen shots  Updated Related Projects section – Split functionality  Updated Copy files section – added disk space validation  Updated Security section - added variable for default profile to ATS\_SystemParameters table |
| 5 Nov 2013 | 2.4 | Updated screen shots  Updated DB model section – removed all groups related tables from Data Model  Updated tables population in all relevant sections  Updated related projects management section  Updated Hierarchy Tree section |

# Solution Architecture

## Overview

The proposed SCD project provides a framework to manage projects across environments (production, test, development) while maintaining multiple versions, content, and execution parameters for every project in any of the environments.

The proposed solution is aimed at addressing the operational requirements that were identified as the highest priority and most business impacting. These include:

* Support multiple versions for every project in every environment
* Organize and catalog content and metadata for every project\version (executable files, other files, execution parameters)
* Support users’ authentication and authorization at an environment and project granularity
* Manage test equipment and work stations – future phases
* Support multiple environments, each with its own projects tree – future phases

Following we will present an overview and the technologies that will be used in the development of the project, and provide a high-level list of proposed modules for the initial phase.

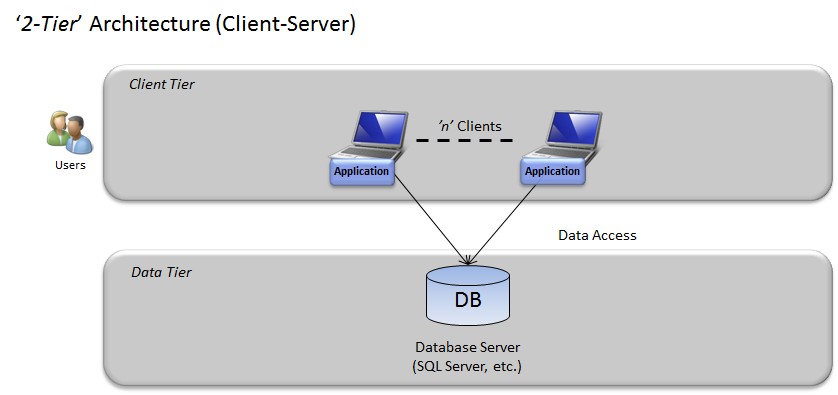
## Architecture and Technology

The proposed solution will be an N-Tiers, N-Layers, Componentized and Object-Oriented Rich Client Application (RCA) as detailed below.

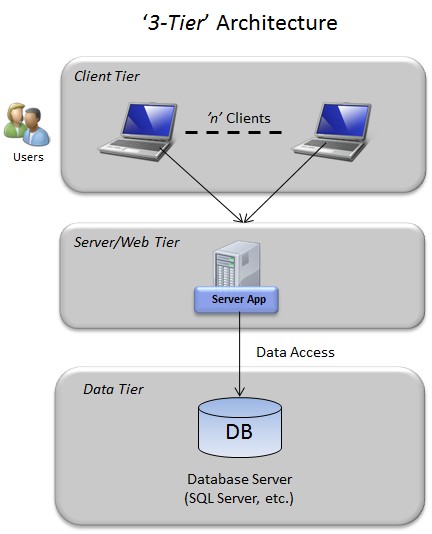
### Physical Architecture (Tiers)

The schemas below present the proposed physical (tiered) architecture for the solution. Being componentized solution, it could be deployed as a 2-Tier (client-server) solution, or as a 3-Tier solution:

* If deployed as a 2-Tier, all components, including Presentation, Business Logic, and Data Access Layer will be deployed on the client’s machines, and only the database will be on a central, separate server.



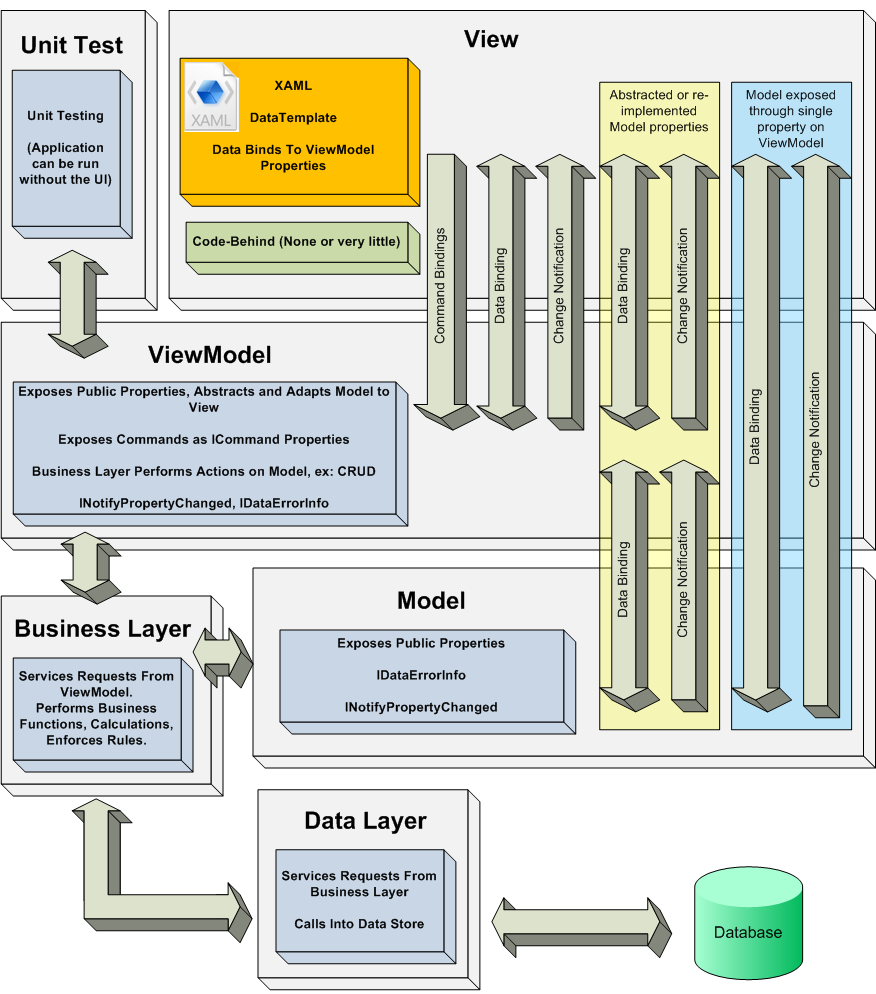
* If deployed as a 3-Tier, the Services Tier will be installed on a separate server. Such architecture can scale in the middle tier (the services tier) to support increased number of users when required.



### Logical Architecture (Layers)

Layers are viewed as logical groups of software components, stacked horizontally, that compose an application or service. They help differentiating between the various types of tasks performed by components, offering a design that maximizes reuse and enhances maintainability while applying the principle of SoC (Separation of Concerns) within the architecture.

The proposed solution will comprise of a WPF (Windows Presentation Foundation) Presentation Tier, developed using the MVVM (Model-View-ViewModel) design pattern. The UI will call WCF (Windows Communication Foundation) services in the Services Tier. These services will communicate with the Data Tier, which will be a SQL Server database. The Services Tier will receive and return Business Objects, which are entities that represent (logically) the data required by the application. The logical architecture of the layers and components in the solution is described below:



### Technologies Used

.NET Framework 4 (Client Profile) and above (for development and runtime)

WCF 4.0 and above (for server-side services layer runtime)

VS 2010 and above (the development environment)

SQL Server 2008 and above (the database)

The system should be operational over Windows XP platforms.

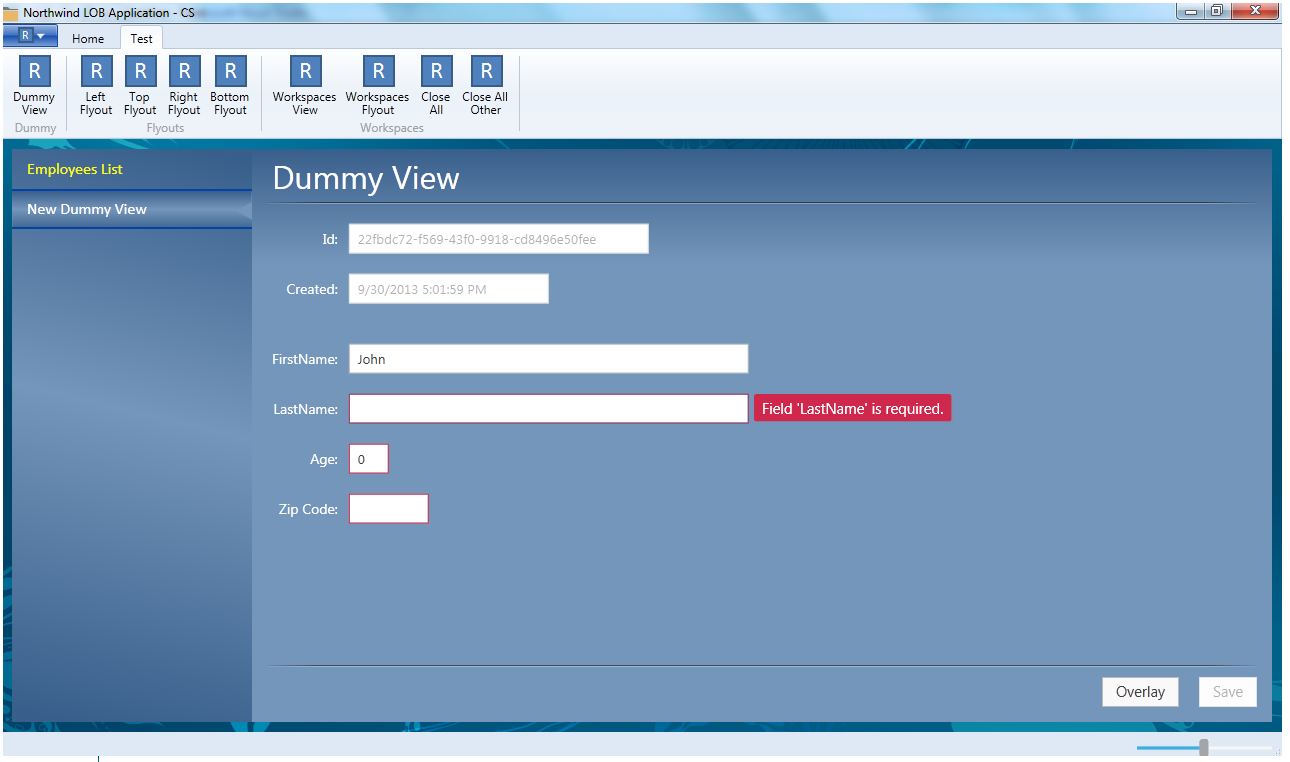
# Solution Components

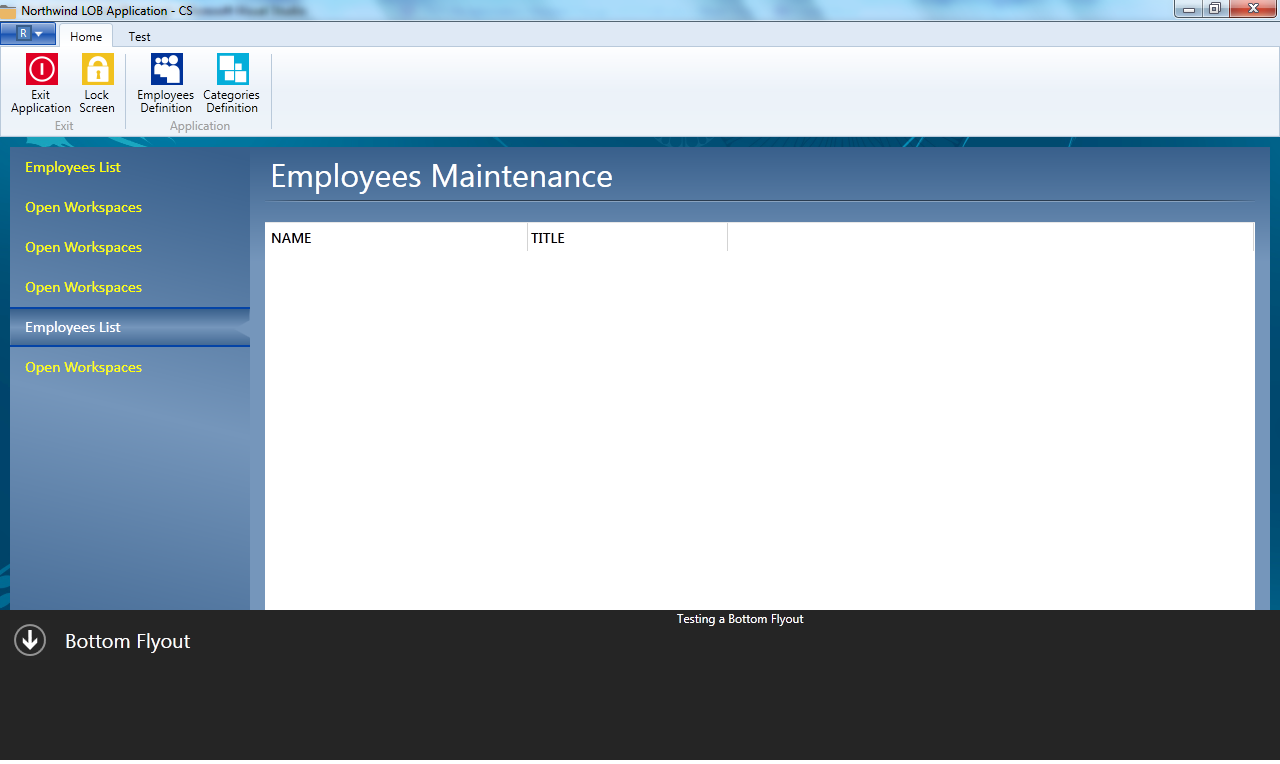
## User Interface Winframe

The system will be developed under a common framework that will launch the various functions and applications, such as the Project Explorer.

Chapter 4, from 4.2 onward, presents winframe screen designs in order to elaborate the various functions and work flows that the system will support.

The following screenshots though give an idea of the system look & feel; the system screens will all adhere to the same look and feel.





## Project Explorer

Enables displaying and traversing thru the projects tree (all projects and versions that live under the root environment).

The project Explorer provides the end users with user friendly run time environment as well as Projects hierarchy review and management.

The various functions, screens and flows are described under Project Management.

## Project Management

Defining projects (organized hierarchically) and linking content to a project. All projects will be managed by version, displayed and maintained in an explorer - like interface that enables traversing up and down the tree (within the environment).

General assumptions:

1. ‘Refresh’ option will be available on a tree level.
2. Control fields are updated when executing any DML statement
   * LastUpdateTime
   * LastUpdateUser
   * LastUpdateComputer
   * LastUpdateApplication
3. LastUpdateTime will be validated prior to each DML statement in order to prevent parallel updates from several instances.

An error message will be issued in case when object was updated since last retrieved:

MessageId 104: “An object has been updated by another user. Please refresh and try again” (OK).

1. Database rollback will be performed in case when transaction has partially failed. In this context transaction is a sequence of DML statements executed upon a single ‘Save’ action performed by the end user.
2. Any DB/Network/API call will be validated for Success/Failure; generic error message will be displayed to the user when unexpected error occurs:

MessageId=105: “Unexpected error occurred. Please try again” (OK).

### Hierarchy Tree – Menus, screens and fields

**Folder**

Folder is a node without any immediate child content. ATS\_Hierarchy.NodeType=’F’ for Folder node.

When user selects a folder node, three tabs will be displayed on the right side of the screen: Folder tab, with listed subfolders, Certificates tabs and Properties tab. Folder right click menu will contain the following options:

New

Cut

Paste – option is displayed only when previous Cut action was performed

Delete – option is displayed only for empty folder with no content ever associated.

DB validation for displaying Delete option:

ATS\_Hierarchy.NodeType=’F’

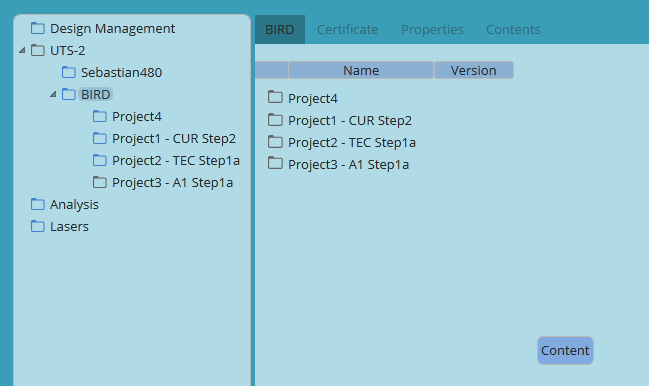
AND

NOT EXISTS (select 1 from ATS\_Hierarchy where ParentId=&selected node id)

Each menu option is a subject for security validation – refer [Security section](#_Security)

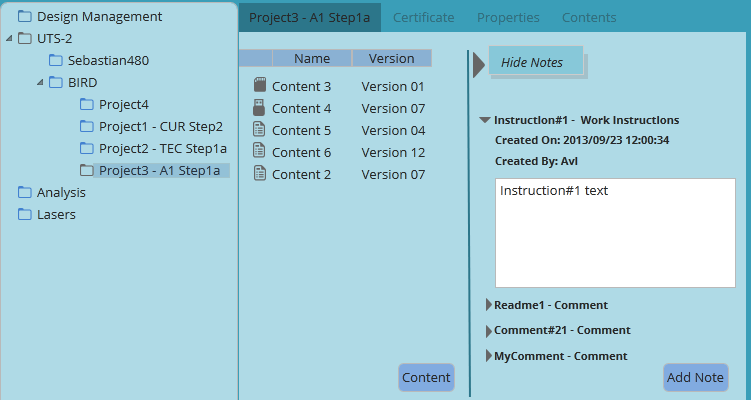
Tree details and properties (parent folder, last update date, folder name, description, etc. – see [Table Definition section](#_Hierarchy)) are stored in ATS\_Hierarchy table.

**Subfolders will be displayed on Folder first tab (similar to Windows files system)**

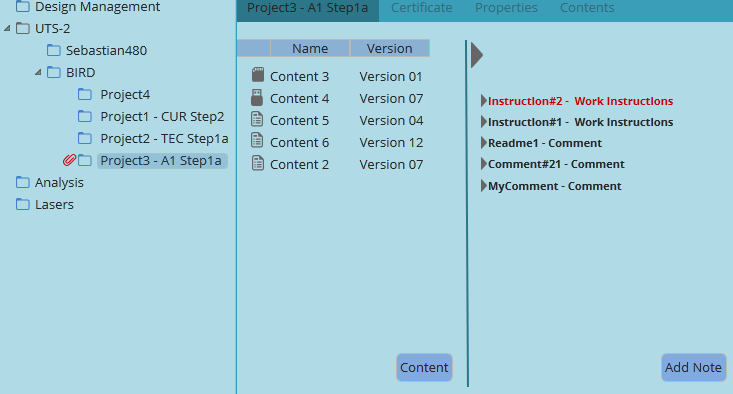


‘Show/Hide Notes’ option will be available on Folder tab.

In case when there is at least one note of type ‘W’ (Work Instructions) associated to the folder, ‘Show Notes’ mode will be the default view for the selected folder.



In case when there is at least one note with ATS\_Note.SpecialInd=’1’ the folder tree view will be distinguished by special icon. Special note title color will be red (ATS\_Note.SpecialInd=1).



Folder notes details are retrieved from ATS\_Note table by HierarchyId=Id of the selected project/folder.

Date – ATS\_Note.CreationDate

Note Type – Comment or Work Instructions, determined by ATS\_Note.NoteType value (‘C’ – Comment, ‘W’ – Work Instructions)

Note title –ATS\_Note.NoteTitle

Note Title right click option:

Disable

Edit

See [Notes Management](#_Notes_management) section.

Disable Note action and disabled notes visibility are subject for security validation, please refer [Security section](#_Security).

Content button – Contents tree is displayed upon pressing the button that activates the Content Management getTreeObject function.

Add Note button – available on the notes section

Save/Cancel button – available when assigning/deleting contents or adding/editing notes.

Adding contents/Notes is a subject for security validation - see details in [Security section](#_Security)

**Certificate tab**

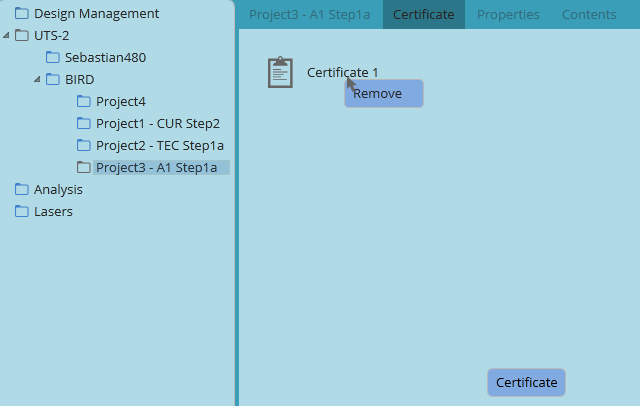
Certificate tab displays certificates associated to selected project/folder.

Folder certificates details are retrieved from ATS\_FolderCertificate table by ATS\_Hierarchy.Id – see [Table Definition section](#_FolderCertificate) ‎.

Certificate name, icon – Open Issue #1 (API – get certificate details by Certificate ID)

Certificate button – certificates list is displayed upon pressing Certificate button (missing API – Open Issue #1)

Certificate is applicable for current folder only.



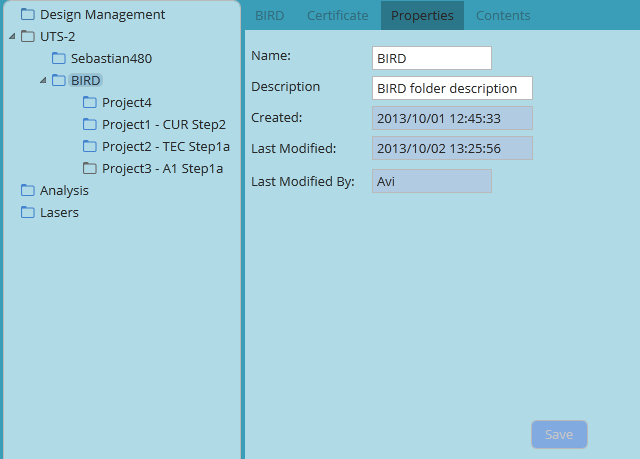
**Properties Tab**

Name: Editable, text 30, ATS\_Hierarchy.Name

Description: Editable, text, ATS\_Hierarchy.Description

Created: Not editable, text, ATS\_Hierarchy.CreationDate

Last Modified & Last Modified By: Not editable, determined by latest LastModifiedDate from all related tables.



**Project**

Project is a folder containing at least one content. Project tree details (parent folder, last update date, folder name and sequence) are retrieved from ATS\_Hierarchy table. ATS\_Hierarchy.NodeType=’P’ for Project node.

Project right click menu will contain the following options:

Show/Hide Versions – according to current view mode

New Clone Project🡪Related, Regular – User will be able to decide whether to create related project (Multi MAKAT) or independent

Cut

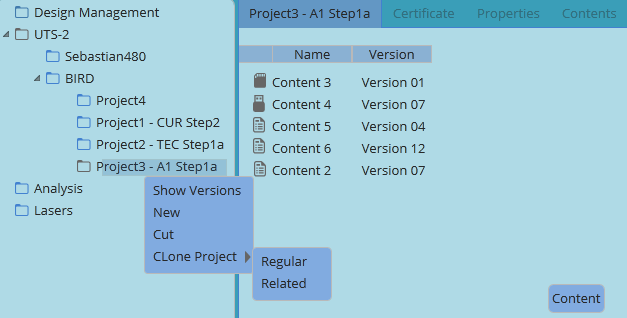
Paste

Disable/Resume – based on current project status (ATS\_Hierarchy.ProjectStatus field)

Project menu options are subject for security validation - see details in [Security section](#_Security)

**Project first tab**

Project first tab will show list of contents associated to the project version



If ‘Show Versions’ option is not selected then version nodes will be hidden on hierarchy (left) side of the screen.

Contents list including content name and content version name will be displayed on the right side of hierarchy window.

User will be able to move Contents up/down in order to update content’s sequence number.

DB update for all contents which sequence number was changed:

UPDATE ATS\_VersionContent

SET ContentSeqNo=’&content sequence number’

WHERE VersionId=’&version id of the selected project’

Content button will be available on the first tab. Contents tree will be displayed to the user when ‘content’ button is selected (API – see [API’s](#_getTreeObjects_function) [section](#_getTreeObjects_function)).

Save/Cancel buttons will be available when user assigned/removed Contents.

Tab fields values:

Name (content name): call getContentDetailsByVersionId (API - Open Issue #1)

Version (content version name): call getContentDetailsByVersionId (API - Open Issue #1)

\*\*\*Content version Id is retrieved from ATS\_VersionContent.ContentVersionId column.

Sequence (Content execution sequence number) 🡪 ATS\_VersionContent.ContentSeqNo

Show/Hide Notes option – see Folder tab [description](#showNotes)

If there are Work Instructions or special notes assigned to the folder, then Project’s first tab will be displayed by default in “Show Notes” mode.

DB:

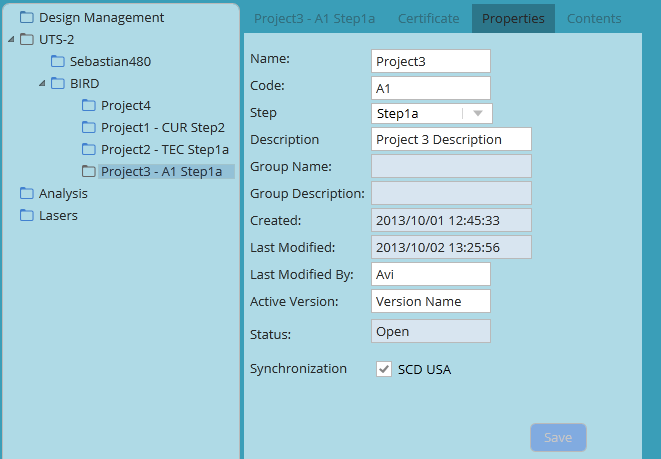
EXISTS (select 1 from ATS\_Note

WHERE (NoteType=’W’ or SpecialInd=1)

AND HierarchyId=’&selected node id’)

**Certificate tab** – see [Folder certificate tab](#Certificate) description.

Project Properties tab



Name: Editable, text 30 (?), ATS\_Hierarchy.Name

Code: Editable, text 20, ATS\_Hierarchy.ProjectCode

Step:

* 1. Regular project

Editable, Drop down list retrieved from ATS\_ProjectStep.StepDescription, populated with ATS\_Hierarchy.ProjectStep by project node Id

* 1. Related project

Editable, Drop down list retrieved from ATS\_ProjectStep.StepDescription, populated with ATS\_Hierarchy.ProjectStep by group node Id:

**SELECT h2.ProjectStep**

**FROM ATS\_Hierarchy AS h1 JOIN**

**ATS\_Hierarchy AS h2 ON h2.Id = h1.GroupId**

**WHERE (h1.Id = ‘&selected project Id’)**

Description: Editable, text, ATS\_Hierarchy.Description

Group Name & Group Description:

if ATS\_Hierarchy.GroupId is null then the fields are hidden

if ATS\_Hierarchy.GroupId is not null then

Group Name: Not Editable, Text 30 (?), ATS\_Hierarchy.Name of the node with Id=GroupId of the selected project

SELECT h2.Name

FROM ATS\_Hierarchy AS h1 JOIN

ATS\_Hierarchy AS h2 ON h2.Id = h1.GroupId

WHERE (h1.Id = ‘&selected project Id’)

Group Description: Not Editable, Text 30 (?), ATS\_Hierarchy.Description

SELECT h2.Description

FROM ATS\_Hierarchy AS h1 JOIN

ATS\_Hierarchy AS h2 ON h2.Id = h1.GroupId

WHERE (h1.Id = ‘&selected project Id’)

Created: Not editable, text, ATS\_Hierarchy.CreationDate

Last Modified & Last Modified By: Not editable, determined by latest LastModifiedDate from all project related tables

Active Version: ATS\_Version.VersionName

Status: Not editable, ATS\_Hierarchy.ProjectStatus

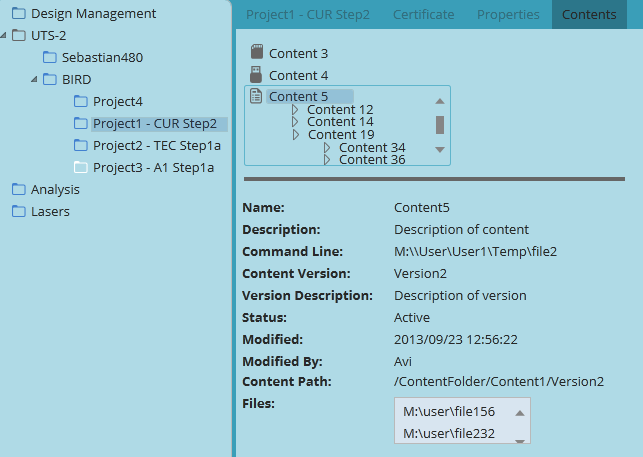
Synchronization: the checkbox is checked if entry exists in ATS\_ProjectSynchronization table and not expired.

When Project Code and Project Step are populated, displayed folder name will be a concatenation of ATS\_Hierarchy. Name+’ – ‘+ATS\_Hierarchy.ProjectCode+ ATS\_Hierarchy.ProjectStep:

“Project3 – A1 Step1a”

**Contents tab – Change Request on top of the project scope, pending SCD’s approval**

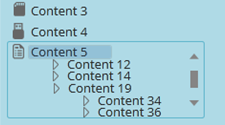
Contents tab is a read only tab where Contents details are displayed.



Content details – read only fields. Call CM function by ATS\_VersionContent.ContentVersionId.

Open Issue #1

Contents will be displayed along with all contents linked to the initial one at any level.



List of content versions linked to parent content version is returned by GetTreeObject function. Last content displayed will not have any content linked to it.

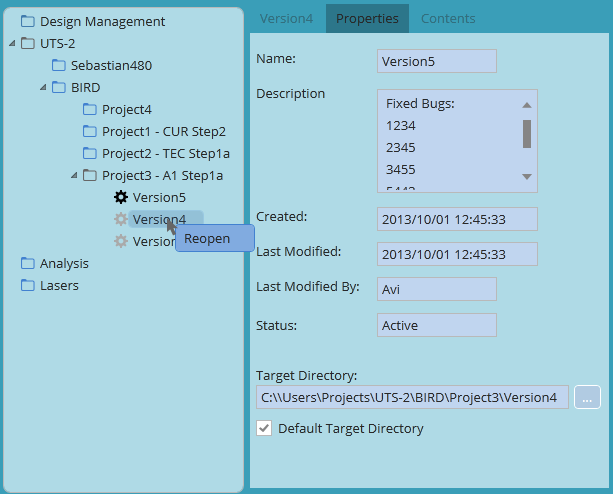
**Version tab** – see [Project tab description](#ProjectTab)

Authorized user will be able to activate contents associated to closed version. Refer [Security](#_Security) section.

Contents tab – see [description](#ContentsTab) in Project section

**Version Properties tab**

All fields on version properties tab are read only.



Regular project versions and versions’ details are retrieved from ATS\_Version table by project node Id:

SELECT VersionId, HierarchyId, VersionName, VersionSeqNo, VersionStatus, Description, TargetPath, CreationDate, LastUpdateTime, LastUpdateUser, LastUpdateComputer,

LastUpdateapplication

FROM ATS\_Version

WHERE (HierarchyId = ‘id of the project’)

Regular project versions and versions’ details are retrieved from ATS\_Version table by GroupId of the selected project:

SELECT VersionId, HierarchyId, VersionName, VersionSeqNo, VersionStatus, Description, TargetPath, CreationDate, LastUpdateTime, LastUpdateUser, LastUpdateComputer,

LastUpdateapplication

FROM ATS\_Version

WHERE (HierarchyId = ‘GroupId of the project’)

Name: ATS\_Version.VersionName

Description: ATS\_Version.VersionDescription

Created: ATS\_Version.CreationDate

Last Modified: ATS\_Version.LastUpdateDate

Last Modified By: ATS\_Version.LastUpdateUser

Status: ATS\_Version.VersionStatus

Target Directory – Text, disabled if Default Target Directory checkbox is checked. Value retrieved from ATS\_Version.TargetPath

Browse button (‘…’) – button, disabled if Default Target Directory checkbox is checked.

Default Target Directory – checkbox, editable

### Folder Management

#### Add New Folder/Project

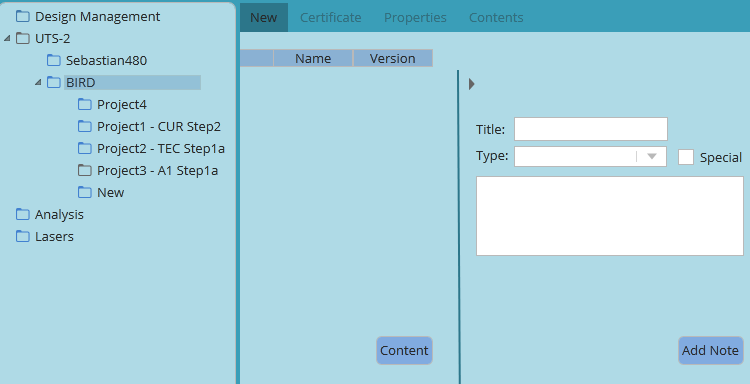
User will be able to add new folder or project by selecting ‘New’ option from folder right click menu.

New project/folder window will be displayed to user and will be similar to the project view screen.

**First tab (New):**

Content button – Contents tree is displayed upon pressing the button, (GetTreeObject). Optional

Add Note button – available on the notes section. Optional



Validations and error handling – refer [Version Management](#_Version_Management) section.

**Certificate tab:**

Certificate button – list of certificates is displayed upon pressing the button, (GerAllCertificates). Optional

Validations and error handling – refer [Assign/Remove](#_Assign/Remove_Certificate_1) Certificate section

**Properties tab:**

Name: Text box, length 30, mandatory

Project Code: Text box, length 10, optional; Available only if content was assigned

Step: Combo box, retrieved from ATS\_ProjectStep table, displays only steps that were not previously created for the project code (unique project code + step), optional. Available only if content was assigned.

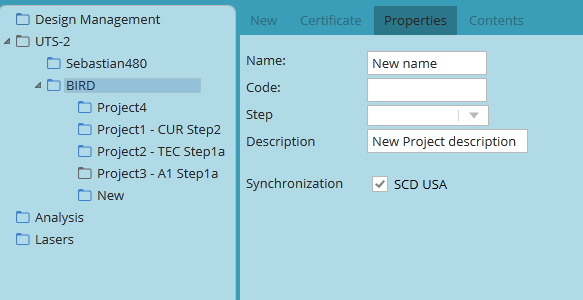
Description: Text, length 100 (?), Mandatory

Synchronization: check list box, values retrieved from ATS\_SyncType.Description column (reference table), optional. Available only if content was assigned.

User will receive an information message upon success:

MessageId 115: “Project '%name' has been created successfully” (OK).

New project node first tab will be displayed to the user when pressing OK.



**Validations:**

* Unique folder (project) name for the parent folder.

If folder name is not unique issue MessageId 100:

“The name is already in use. Please specify another name.” (OK)

* Unique Project Code + Project Step combination, or null

When not unique issue error:

Message 110:

“Project Code and Step combination is not unique. Please update and try again.” (OK)

* O/L – mandatory fields: Name, Version Name, Version Description, mandatory Code when step is populated

**DB**:

ATS\_Hierarchy

|  |  |
| --- | --- |
| Id | sequence |
| ParentId | Parent Id of the folder |
| Sequence | count (child folders) +1 for the parent id |
| NodeType | ‘F’ – if no content was added. ‘P’ – if content was added |
| Name | GUI |
| Description | GUI |
| GroupId | null |
| ProjectCode | GUI |
| ProjectStep | GUI |
| ProjectStatus | If NodeType=‘F’ then null, if NodeType=’P’ then ‘O’ |
| CreationDate | sysdatetime |
| LastUpdateTime | sysdatetime |
| LastUpdateUser | userId – received from access management upon login |
| LastUpdateComputer | Computer Name |
| LastUpdateApplication | PE |







ATS\_Version (only if content was added)

|  |  |
| --- | --- |
| VersionId | Sequence |
| HierarchyId | ATS\_Hierarchy.Id |
| VersionName | GUI |
| VersionSeqNo | 1 |
| VersionStatus | ‘O’ |
| Description | GUI |
| TargetPath | GUI |
| CreationDate | sysdatetime |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateapplication |  |

ATS\_VersionContent (only if content was added)

|  |  |
| --- | --- |
| Rowid | sequence |
| VersionId | ATS\_Version.VersionId |
| ContentVersionId | ContentVersionId – GUI, API |
| ContentSeqNo | Content sequence number - GUI |
| CreationDate | Sysdatetime |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateApplication |  |

ATS\_ProjectSynchronization (for checked Sync types)

|  |  |
| --- | --- |
| Rowid | sequence |
| HierarchyId | ATS\_Hierarchy.Id |
| SyncType | GUI |
| EffectiveDate | sysdatetime |
| ExpirationDate | null |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateapplication |  |

FolderCertificate – see [Assign/Remove Certificate](#_Assign/Remove_Certificate) section.

Note – see [Notes Management](#_Notes_management) section.

#### Clone Project, Cut and Paste

* User will be able to move (cut and paste) any branch of the hierarchy tree.

The business logic and implementation is similar to Windows files system.

DB: ATS\_Hierarchy.ParentId of moved folder ATS\_Hierarchy.NodeId of the new parent folder

ATS\_Hierarchy.Sequence of moved folder will be set to N, where N is total number of child folders including the new one.

User will be able to move folder by drag & drop. In this case moved/copied folder sequence number will be determined by position within the destination branch.

**Validations**:

Unique folder name for the parent folder

If folder name is not unique issue MessageId 100:

“The name is already in use. Please specify another name.” (OK)

* Right click🡪Clone Project🡪Regular/Related options will be available for Project folder only.

Subfolders and notes, if exist, are not copied as a part of cloned project.

In case when Project Code and Project Step values are populated for the source project, the user will be requested to set new values for cloned project or remove existing values.

* Clone Project🡪Related - Please see [Related Project Management](#_Create_Related_Project) section for details.
* Clone Project🡪Regular:

Clone Regular project - first tab:

Contents button will be available. User will be able to remove contents inherited from copied project or assign new contents

Clone Regular project - Certificate tab:

Certificate button will be available. User will be able to remove certificates inherited from copied project or assign new certificates.

Clone Regular project - Properties tab

All displayed values are inherited from copied project.

Name – editable, mandatory.

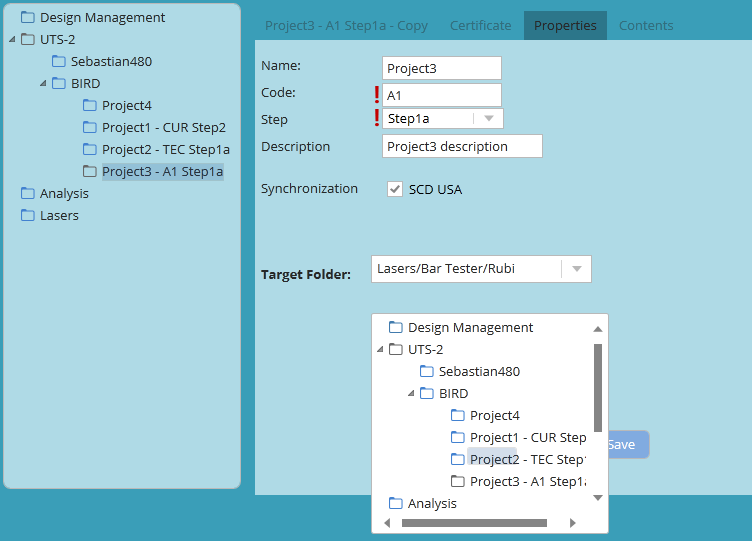
Code, Step – editable, mandatory to update if populated or set both null.

Description – editable, mandatory (not null).

Target Folder – tree combo box, editable, mandatory.

Synchronization: editable, optional.

Save option will be available once Project Code and Project Step fields are updated.



User will receive an information message upon success:

MessageId 115: “Project '%name' has been created successfully” (OK).

New project node first tab will be displayed to the user when pressing OK.

DB:

Refer [Add New Project section](#_Add_New_Project) for tables’ population.

Refer [Add New Project section](#_Add_New_Project) for validations and error handling.

#### Rename

User will be able to rename folder by updating ‘Name’ field on folder Properties tab

DB: update ATS\_Hierarchy.Name

Validations:

Verify unique folder names for the same parent folder.

If folder name is not unique issue MessageId 100:

“The name is already in use. Please specify another name.” (OK).

#### Delete Folder

Delete option will be available for empty folder with no content ever associated.

DB validation for displaying Delete option:

ATS\_Hierarchy.NodeType=’F’

AND

NOT EXISTS (select 1 from ATS\_Hierarchy where ParentId=&selected node id)

User will receive a warning when trying to delete folder:

Message 101 “Are you sure you want to remove the folder?” (Yes, Cancel)

**DB**:

Delete ATS\_Hierarchy where Id=’%selected node Id’

#### Disable/Resume Project

* User will be able to disable project by selecting ‘Disable’ option from project node right click menu. Disabled project cannot be updated, modified or activated by default user.

Super user will have permission to perform any action on disabled project.

Refer [Security](#_Security) section for details.

Disabled project will be distinguished by icon or font on hierarchy tree.

DB:

update Hierarchy set ATS\_Hierarchy.ProjectStatus=’D’ where ATS\_Hierarchy.Id=&selected node id

* User will be able to resume disabled project by selecting ‘Resume’ option from project node right click menu
* **DB**:

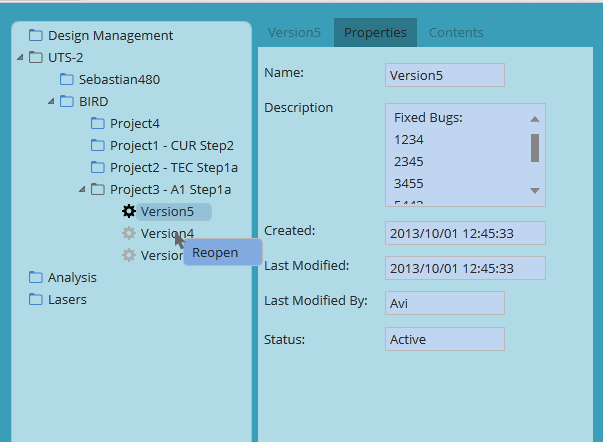
Update Hierarchy set ATS\_Hierarchy.ProjectStatus=’O’ where ATS\_Hierarchy.Id=&selected node id

#### Show/Hide Versions

User will be able to select versions view mode by switching between Show/Hide Versions right click menu.

List of contents assigned to the active version will be the default view on project’s first tab when ’Show Versions’ mode is not selected. In this case versions nodes and details will not be displayed.

All active and historical versions and version tabs will be displayed as a part of hierarchy tree when user selects ‘Show Versions’ option. Versions will be distinguished by icon/font based on version status (active /closed).



**DB**:

Versions properties: ATS\_Version table

Version contents (contentVersionId key): ATS\_VersionContent table

Contents details: API, GetTreeObject. Open Issue #1

#### Related Project Management

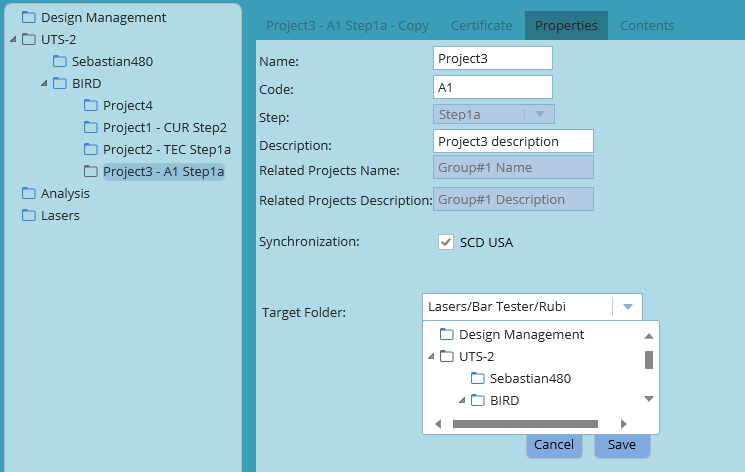
User will be able to create related project by selecting Clone Project🡪Related option from parent project right click menu. Related project will inherit Versions and all versions properties, Notes and Project step from parent project.

User will be able to update project Name and project Code, certificates, Description, synchronization type and to select a target folder from a tree combo box.

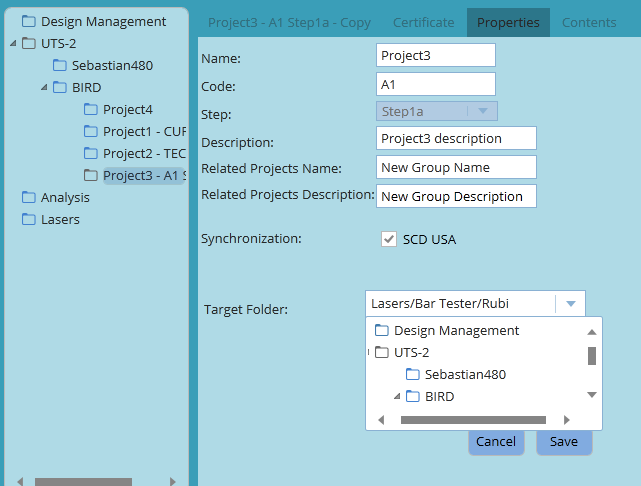
If parent project already belongs to the group, then Related Group Name and Related Group Description fields will be populated with group name and description and will not be editable. Otherwise user will be required to specify new group name and description.

Project Step, Notes and contents tab are not editable for new related project.

Existing Related Group:



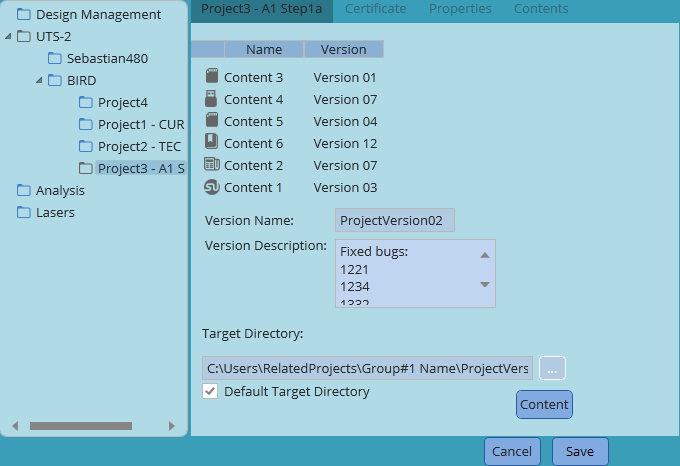
New related group:



User will be able to update Default Target directory for active version of new Related Group.

Target directories for closed versions will be set to default.

Version default target directory value is a concatenation of ‘RelatedGroupLocalPath’ value + Group Name + Version Name.



Validations:

* + Unique folder name (Project name) for the Target Folder
  + Unique Project code + Project Step
  + Unique Related Projects group name

In case when Group Name is not unique issue error:

Message 123: ‘Related projects Group Name is already in use. Please specify another name’

* + O/L – all editable fields are mandatory

Error handling - refer [Add New Project section](#_Add_New_Project)

**DB**:

Initial project is Regular (New Group)

ATS\_Hierarchy, Group node

|  |  |
| --- | --- |
| Id | sequence |
| ParentId | null |
| Sequence | 0 |
| NodeType | ‘G’ |
| Name | GUI, Group Name |
| Description | GUI, Group Description |
| GroupId | null |
| ProjectCode | null |
| ProjectStep | GUI |
| ProjectStatus | null |
| CreationDate | sysdatetime |
| LastUpdateTime | sysdatetime |
| LastUpdateUser | userId – received from access management upon login |
| LastUpdateComputer | Computer Name |
| LastUpdateApplication | PE |

ATS\_Hierarchy, Project node

|  |  |
| --- | --- |
| Id | sequence |
| ParentId | Id of the Target folder (GUI) |
| Sequence | count (child folders) +1 for the parent id |
| NodeType | ‘P’ |
| Name | GUI |
| Description | GUI |
| GroupId | Id of the previously created group |
| ProjectCode | GUI |
| ProjectStep | null |
| ProjectStatus | ‘O’ |
| CreationDate | sysdatetime |
| LastUpdateTime | sysdatetime |
| LastUpdateUser | userId – received from access management upon login |
| LastUpdateComputer | Computer Name |
| LastUpdateApplication | PE |



ATS\_ProjectSynchronization (for checked Sync types)

|  |  |
| --- | --- |
| Rowid | sequence |
| HierarchyId | ATS\_Hierarchy.Id |
| SyncType | GUI |
| EffectiveDate | sysdatetime |
| ExpirationDate | null |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateapplication |  |



Update ATS\_Hierarchy, parent project entry:

UPDATE ATS\_Hierarchy set GroupId=’id of the created group’ where Id=’id of the parent project’

Update ATS\_Version:

* link parent project’s versions to group node

UPDATE ATS\_Version set HierarchyId=’id of the created group’

where HierarchyId =’id of the parent project’

* Update target path for closed versions:

UPDATE ATS\_Version set TargetPath=ATS\_SystemParameters.RelatedProjectLocalPath+Group Name+Version Name

Where VersionStatus=’C’

* Update target path for active version

UPDATE ATS\_Version set TargetPath=GUI Value

Where VersionStatus=’A’

Update ATS\_Note, link parent project’s notes to Group Id:

UPDATE ATS\_Note set HierarchyId=’id of the created group’

where HierarchyId =’id of the parent project’

Initial project is Related (Existing Group)

ATS\_Hierarchy, Project node

|  |  |
| --- | --- |
| Id | sequence |
| ParentId | Id of the Target folder (GUI) |
| Sequence | count (child folders) +1 for the parent id |
| NodeType | ‘P’ |
| Name | GUI |
| Description | GUI |
| GroupId | Id of the group |
| ProjectCode | GUI |
| ProjectStep | null |
| ProjectStatus | ‘O’ |
| CreationDate | sysdatetime |
| LastUpdateTime | sysdatetime |
| LastUpdateUser | userId – received from access management upon login |
| LastUpdateComputer | Computer Name |
| LastUpdateApplication | PE |

ATS\_ProjectSynchronization (for checked Sync types)

|  |  |
| --- | --- |
| Rowid | sequence |
| HierarchyId | ATS\_Hierarchy.Id |
| SyncType | GUI |
| EffectiveDate | sysdatetime |
| ExpirationDate | null |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateapplication |  |

ATS\_FolderCertificate – see [Assign/Remove Certificate](#_Assign/Remove_Certificate) section

User will receive an information message upon success:

MessageId 116: “New related project ‘%name’ has been successfully created and assigned to the group %GroupName” (OK).

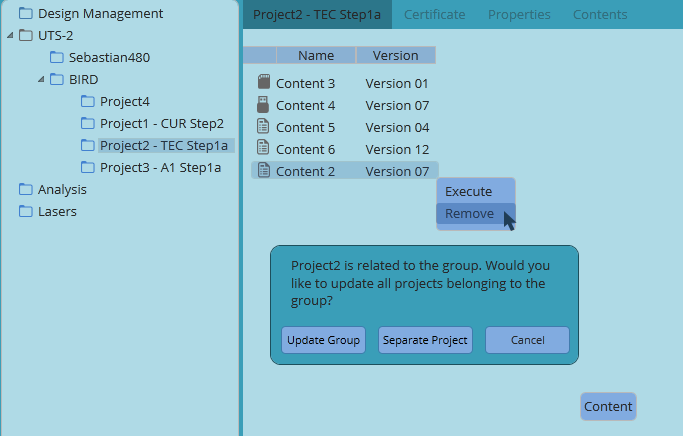
New project node first tab will be displayed to the user when pressing OK.

* **Update one of the related projects:**

User will selects one of the following options when modifying project belonging to group (add/remove contents, manage notes, edit Project Step, edit version target directory):

* + Separate project from the group
  + Update all projects belonging to the group
  + Cancel update

Message 111: “Project '%name' is related to the group. Would you like to update all projects belonging to the group?”



* Cancelling the update – return to previous screen, no action
* Update all projects belonging to the group:
* Manage Versions (add/remove content, change target directory): update ATS\_Version/ATS\_VersionContent tables, ATS\_Version.HierarchyId= GroupId of the selected project , see [Version Management](#_Version_Management) section.
* Edit ProjectStep: Update ATS\_Hierarchy set ProjectStep for Id=GroupId of the selected project.
* Manage Notes: Update ATS\_Note table, HierarchyId=GroupId of the selected project, see [Notes Management](#_Notes_management) section

User will receive an information message upon success:

MessageId 117: “Related projects group has been successfully updated.” (OK).

* Separate project from the group:

**DB**







Version – all values for all versions are inherited from the initial group or from GUI if updated

|  |  |
| --- | --- |
| VersionId | Sequence |
| HierarchyId | Id of the selected project |
| VersionName | Inherited from the initial group or from GUI if updated |
| VersionSeqNo | Inherited from the initial group or from GUI if updated |
| VersionStatus | Inherited from the initial group or from GUI if updated |
| Description | Inherited from the initial group or from GUI if updated |
| TargetPath | Default or from GUI if updated |
| CreationDate | sysdatetime |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateapplication |  |

ATS\_VersionContent - all values for all versions are inherited from the initial group,

or from GUI if updated

|  |  |
| --- | --- |
| Rowid | sequence |
| VersionId | ATS\_Version.VersionId |
| ContentVersionId | Inherited from the initial group or from GUI if updated |
| ContentSeqNo | Inherited from the initial group or from GUI if updated |
| CreationDate | Sysdatetime |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateApplication |  |



Note – inherited from the initial group or from GUI if updated:

|  |  |
| --- | --- |
| Id | sequence |
| HierarchyId | Id of the selected project |
| NoteType | inherited from the initial group |
| NoteStatus | inherited from the initial group or from GUI if updated |
| NoteTitle | inherited from the initial group or from GUI if updated |
| NoteText | inherited from the initial group or from GUI if updated |
| SpecialInd | inherited from the initial group or from GUI if updated |
| CreatedByUser | inherited from the initial group |
| CreationDate | sysdatetime |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateapplication |  |



Update ATS\_Hierarchy set GroupId null for the selected project.

Update Hierarchy set ProjectStep=’group ProjectStep’ where id=’selected project id’

Verify that 2 or more projects left in the group:

Select count(Id) from ATS\_Hierarchy where GroupId=id of the group.

In case when only one project left in the group, perform the following action for the project:

Update ATS\_Hierarchy set GroupId null, ProjectStep is populated with Group record value if any.

Update ATS\_Version set HierarchyId to id of the project where HierarchyId=Id of the Group , set TargetPath to default (all Group versions are updated)

Update ATS\_Note set HierarchyId to id of the project where HierarchyId=Id of the Group (all Group notes are updated).

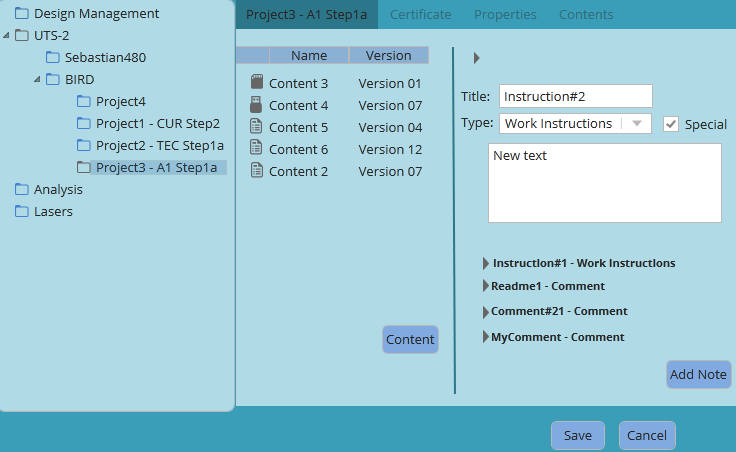
Updated project node first tab will be displayed to the user when pressing OK.

Error Handling – refer [Add New Project](#_Add_New_Project) section

Updating related project is a subject for security validation – refer [Security](#_Security) section.

#### Notes management

* + User will be able to assign note to any folder by pressing ‘Add Note’ button on the notes tab.



Type – NoteType.Description, drop down list, mandatory

Title – text, 50, mandatory

Note text: text, 500, mandatory

Special – Indicator, optional

Add Note, Cancel, Save – buttons

**Validations**:

* O/L - Save button will be disabled unless all fields are populated correctly.
* In case when handled node is a project - verify that the project doesn’t belong to a group of Related Projects:

If ATS\_Hierarchy.GroupId is null then user will select between the following options:

* + Separate one or more related projects from the group
  + Update all projects belonging to the group
  + Cancel update

Add/Update Related Projects notes – see details in [Related Project Management](#_Create_Related_Project) section.



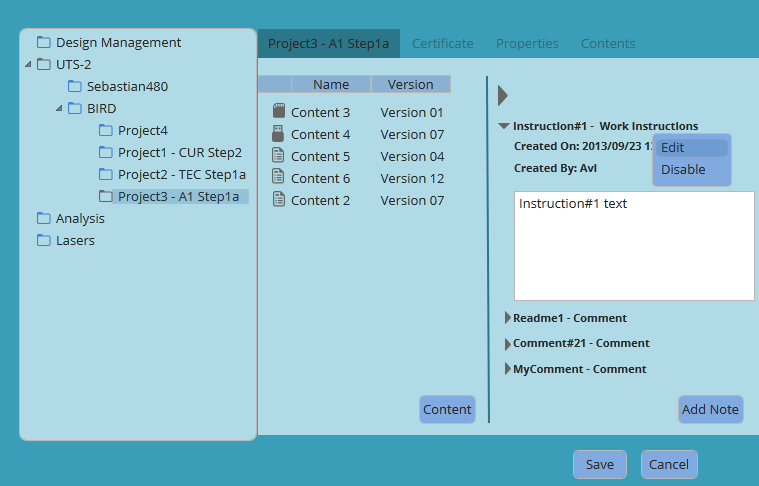
DB:

Note

|  |  |
| --- | --- |
| Id | sequence |
| HierarchyId | Id of the selected project |
| NoteType | GUI |
| NoteStatus | ‘A’ |
| NoteTitle | GUI |
| NoteText | GUI |
| SpecialInd | GUI (0,1) |
| CreatedByUser | GUI |
| CreationDate | sysdatetime |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateapplication |  |



* User will be able to edit existing note. ‘Edit’ option will be available in the right click menu. Cancel and Save buttons will be available once Edit button is pressed.



**Validations**:

* In case when handled node is a project - verify that the project doesn’t belong to a group of Related Projects:

If ATS\_Hierarchy.GroupId is not null then refer [Related Project Management](#_Create_Related_Project) section.

**DB**:

Update Note set ATS\_Note.NoteText =’&new text’

where ATS\_Note.Id=’& selected note id’

* User will be able to disable a note by selecting a ‘Disable’ option from note title right click menu.

A warning message will be displayed upon selecting this option:

Message 102: “Are you sure you want to disable the note?” (Yes, Cancel).

DB:

Update ATS\_Note set NoteStatus=’D’ where id=selected note id

Add/Edit note actions are subject for security validation – refer [Security](#_Security) section.

#### Modify Folder/Project

User will be able Modify folder/project by selecting a node.

The first tab (Folder/Project name), Certificate and Properties tabs will be available for editing.

* Edit Name/Description/ProjectCode

DB:

Update Hierarchy set Name/Description/ProjectCode/ProjectStep for selected node Id

* Edit Synchronization

DB:

Insert into ProjectSynchronization values for checked SyncType

Or

Update ProjectSynchronization set ExpirationDate for unchecked SyncType

* User will be able to assign/delete contents to existing project/folder – see [Version Management](#_Version_Management) section .
* User will be able to assign/remove certificates to existing project/folder – see [certificates](#_Assign/Remove_Certificate)  section

Error Handling – refer [Add New Project](#_Add_New_Project) section

### Version Management

* Create New Version

Content button will be available on the first tab of any folder. Contents tree will be displayed to the user when ‘content’ button is selected (API – see [API's section](#_getTreeObjects_function)).

User will be able to drag & drop content versions from the contents tree and remove existing contents.

Description and Name fields will be available once user assigns new or removes old Content.

Fields:

Description – text, 200, mandatory

Name – text, 30, mandatory

Target Directory – Text, Mandatory, disabled if ‘Default Target Directory’ checkbox is checked. The field is auto populated with default value.

Browse button (‘…’) - disabled if ‘Default Target Directory’ checkbox is checked.

Default Target Directory - check box, enabled after Project Name and Version Name fields are populated.

Default Target Directory value:

‘ProjectLocalPath’ value + Hierarchy full path of the project + Project Name + Version Name

‘ProjectLocalPath’ value retrieved from ATS\_SystemParameters table:

select value from ATS\_SystemParameters where variable=’ProjectLocalPath’

Example:

ProjectLocalPath=’ **C:\Users\Projects\**’

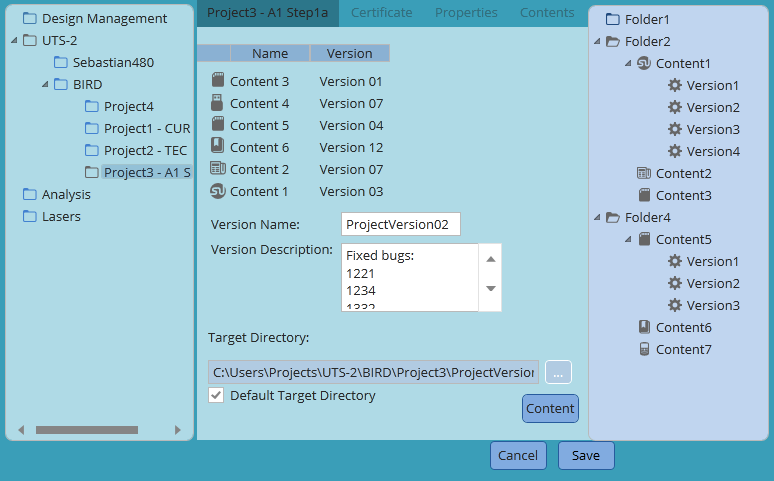
project hierarchy full path is UTS-2\BIRD\

Project Name is Project3

Version Name is Version02

then default target directory for Project3 Version02 is

‘**C:\Users\Projects\**UTS-2\BIRD\Project3\Version02’



Save button is available once all mandatory fields are populated correctly.

User will be able to save new project version or cancel action.

New version will be created upon completing the action (Save).

**Validations**:

* Unique version name for the project.

If version name is not unique issue error:

Message 124: “Version Name is already in use. Please specify another name.”

* Unique contentId for project version – API, validate ContentId by contentVersionId. (get contentId for each contentVersionId, validate uniqueness – GetTreeObject). ContentId validation includes recursive validation for all linked contents.

Error handling:

If user has added two or more content versions of the same content issue error:

Message 108: “Two or more Content versions of the same Content have been added. Please remove and try again”. (OK)

* If user has added content that is linked to another previously added content issue error:

Message 119: “Content %name1 is referenced by content %name2. Please remove and try again.”

The validations above performed twice: first time when user assigns content, per each new content assigned and second time upon save.

* If user has added two contents linking the same content issue error:

Message 122: “Content %name1 and content %name2 are referenced by content %name3. Please remove %name1 or %name2 and try again.”

The validations above performed twice: first time when user assigns content, per each new content assigned and second time upon save.

* If content status is ‘Retired’ then user will receive a warning message:

Message 120: “Content %name is retired. Do you want to assign retired content to the project?” (Yes, Cancel)

* Unauthorized user will not be able to assign retired content to project.

Error message will be received when performing the action (press ‘Yes’ on message 120):

MessageId 106: “User %name is not authorized to perform this action” (OK)

* O/L – Mandatory fields – Version Name, Version Description, Target Directory
* Verify that the project does not belong to group of related projects.

If ATS\_Hierarchy.GroupId is not null , then refer [Related Projects Management](#_Create_Related_Project) section.

User will receive a warning message upon pressing ‘Save’ button:

MessageId 118: “Are you sure you want to create a new version?” (Yes, Cancel)

Project first tab will be displayed to the user when pressing ‘Yes’ on the message tab.

Initial screen will be displayed if user decided to cancel action.

**DB**:

Creating new project version, new project – see [Add New Project](#FirstVersion) section.

Create new project version, existing project.

ATS\_Version table – old version:

Update ATS\_Version set VersionStatus=’C’, expirationDate=sysdate

Where versionId=’&updated version id’

New Version

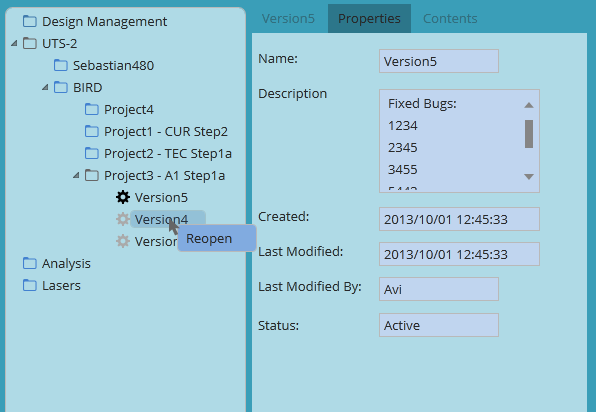
|  |  |
| --- | --- |
| VersionId | Sequence |
| HierarchyId | Id of the project |
| VersionName | GUI |
| VersionSeqNo | Max value +1 |
| VersionStatus | ‘O’ |
| Description | GUI |
| TargetPath | GUI |
| CreationDate | sysdatetime |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateapplication |  |

ATS\_VersionContent – for all contents on the project tab

|  |  |
| --- | --- |
| Rowid | sequence |
| VersionId | ATS\_Version.VersionId |
| ContentVersionId | ContentVersionId – GUI, API |
| ContentSeqNo | Content sequence number - GUI |
| CreationDate | Sysdatetime |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateApplication |  |

* Reopen closed version

User will be able to reopen closed version by selecting ‘Reopen’ option from the right click menu. The option will be available for closed versions only (where ATS\_Version.VersionStatus=’C’).



Current active version will be automatically closed.

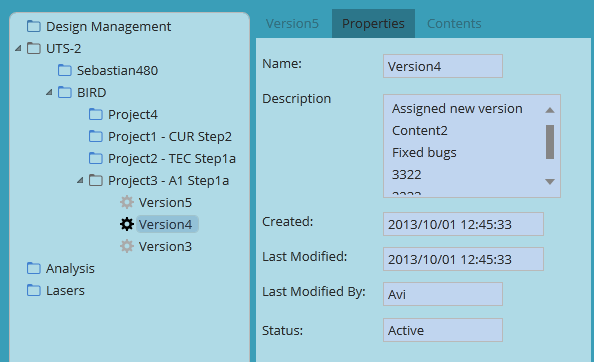
User will receive a warning message when reopening closed version:

Message 112 “Version %[active version name] will be closed. Are you sure you want to reopen version %name?”

**DB**:

Update Version set VersionStatus=’C’, expirationDate= sysdatetime for current active version.

Update Version set VersionStatus=’O’, ExpirationDate null for versionId of reopened version.



## Activate Content and Copy content files

User will be able to activate any content by opening project’s first tab and selecting an ‘Execute’ option from content menu or by double click on Content.

Executable file path is provided as part of content details list by CM function (GetTreeObjects function). The file will be executed from the local drive after being copied (see below). The system will be able to pass parameters (ProjectCode, ProjectStep, etc.) to the executable.

Authorized user will be able to activate content associated to any version regardless the status (Active, Closed) – refer [Security](#_Security) section for details.

Files assigned to project’s contents will be copied from network drive to local machine prior to project activation.

User action: select any content assigned to the project, right click🡪Execute or Double Click on Content.

Each Content file is handled as a concatenation string of fileRelativePath and FIleName values returned by GetTreeObjects function:

**’\’** +‘TreeNode->TreeNodetype-> ContentVersion->ContentFile->**FileRelativePath**’ + ‘**\**’+’ TreeNode->TreeNodetype-> ContentVersion->ContentFile->**FileName**’

The files will be copied to version home folder (ATS\_Version.TargetPath) including FileRelativePath and according to the following rules:

* The system will copy files associated to all contents assigned to the project
* The system will check whether there is available disk space on local machine. The value for required disk space is retrieved from ATS\_SystemParameters table:
* select value from ATS\_SystemParameters where variable=’RequiredDiskSpace’
* When two or more contents source folders (on network drive) contain files with the same name, then only file belonging to the content with the highest sequence number (table driven - ATS\_VersionContent.ContentSeqNo) will be copied to local drive.
* If target destination (on local drive) contains file with the same **FileRelativePath** + name and “Modified” date is equal to the “Modified” date of the source file then it will not be copied.
* If target destination contains file that does not exist in the source location then it will be deleted from target destination.

A progress bar will be displayed while files’ copying is in progress.

Validations:

* Security validation, refer to [Security Section](#_Security)
* Validate that current Workstation is certified for executing the project if applicable:
* Verify that current project requires certification:

Select certificateId from FolderCertificate

where HierarchyId=’&selected node Id’

and ExpirationDate is null

If no row selected then allow action

If at least one row selected then perform additional validations:

* Call getContentDetailsByContentVersionId API (Open Issue #1), validate UncertifiedInd. If ‘Y’ then allow content execution. If ‘N’, then get Workstation certificates:
* Call getCertificateByWorkstationId API (Open Issue #1).

If project certificates (i) are part of (iii) output, then allow action. If one or more project certificates are missing then disable ‘Activate’ option. User tip will be displayed to the user:

MessageId 107: “Current workstation is not certified for the project. The following certificates are missing: %list of missing certificates. Would you like to proceed anyway?” (Yes, Cancel)

Executing project from uncertified workstation is a subject for security validation. Refer [Security](#_Security) section

* If there is no enough disk space issue error message:

Message 121: “There is not enough disk space. Please free disk space and try again.”

## Assign/Remove Certificate

‘Certificate’ button will be available on the Certificate tab of the folder. User will receive list of available certificates upon pressing the button (API – see [API's section](#_getAllCertificates_function)).

User will be able to assign certificate to any folder by Drag & Drop to Certificate tab.

User will be able to remove certificate by selecting a Remove option from certificate right click menu.

User will receive a warning message when selecting ‘Remove’ option from the menu.

MessageId 103: “Are you sure you want to remove the certificate?” (Yes, Cancel)

Validations:

* Unique certificateId for the folder

Error handling:

If two certificates with the same certificateId have been added issue error.

Message 109: “Two or more certificates with the same Certificate ID have been added. Please remove and try again.” (OK)

The validation will be performed twice: first time when user assigns certificate and second time upon save.

**DB**:

ATS\_FolderCertificate, assign certificate:

|  |  |
| --- | --- |
| HierarchyId | NodeId of the folder – ATS\_Hierarchy.NodeId |
| CertificateId | Certificate Id |
| EffectiveDate | sysdate |
| ExpirationDate | null |
| LastUpdateTime | Control fields |
| LastUpdateUser |  |
| LastUpdateComputer |  |
| LastUpdateApplication |  |

ATS\_FolderCertificate, remove certificate:

Update FolderCertificate set expirationDate for selected NodeId and removed CertificateId

## Search

Search option will be available on the tool bar.

User will be able to insert free text or “\*” in the search text box and select one of the following search categories:

Content Name – API (getTreeObjects, map contentVersionId to content name)

Project Code – ATS\_Hierarchy.ProjectCode

Project Description – ATS\_Hierarchy.Description

Folder Name – ATS\_Hierarchy.Name

Notes – ATS\_Note.NoteText

LastUpdateUser – in all tables

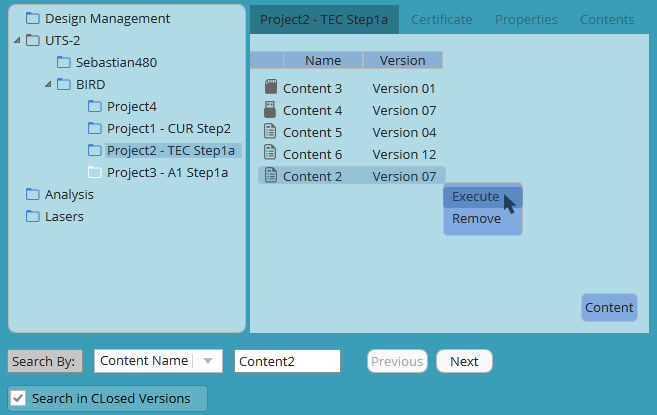
All Categories – union all of the above

Search by – combo box.

Search text box – length 30.

Search in closed Versions – check box, default is false.

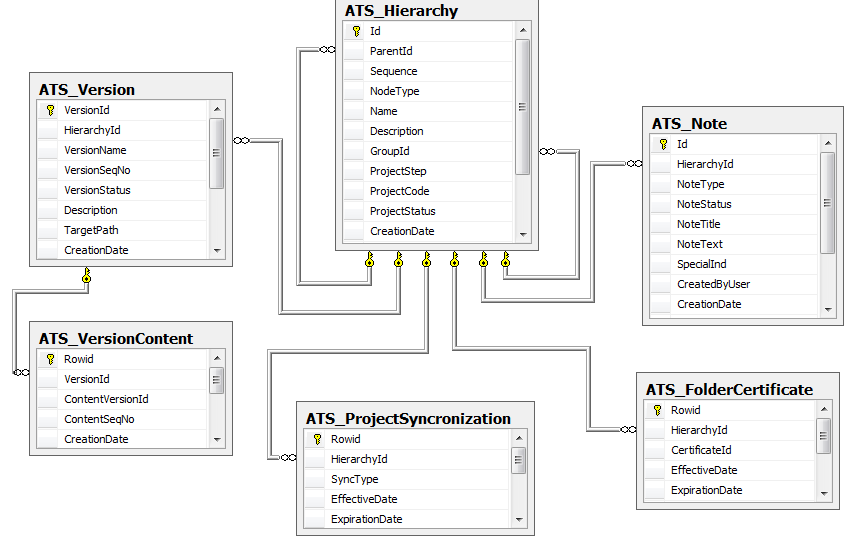
First node that matches the search criteria will be highlighted; ‘Next’ button will be available for viewing additional search results.



# Data Model

## Project Explorer

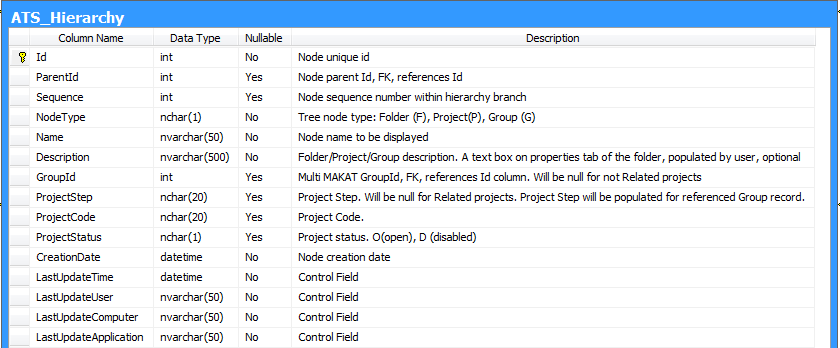
### ERD



### Table Definition

#### ATS\_Hierarchy

Hierarchy tree definition



Id – PK. Hierarchy Id of Project/Folder

ParentId – FK, references Id

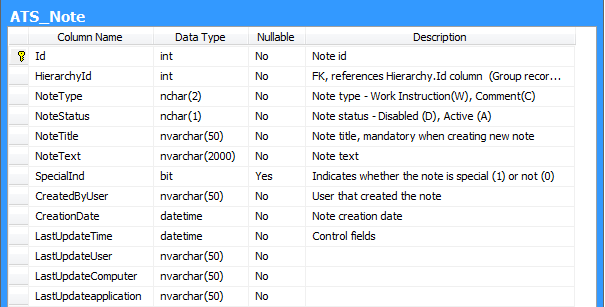
GroupId – FK, references Id

Sequence – unique for the same ParentId, check constraint

Name - unique for the same ParentId, check constraint

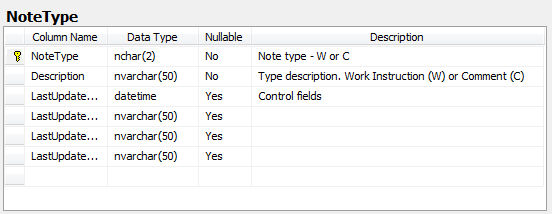
#### ATS\_Note

Note table stores all notes. Each note is associated to one Note Group.



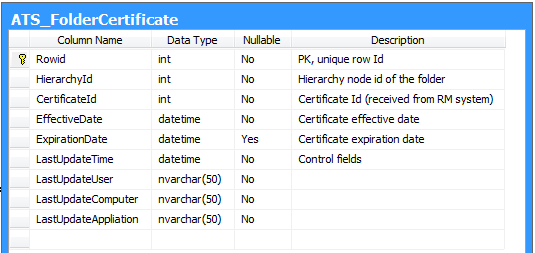
#### ATS\_NoteType

Reference table, describes various note types.



#### ATS\_FolderCertificate

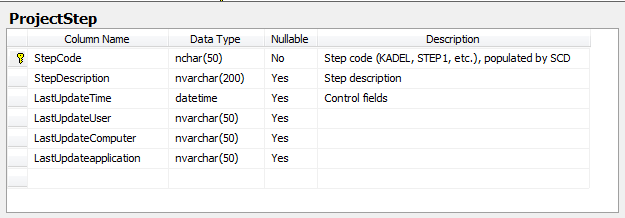
Certificates assigned to the hierarchy node.



HierarchyId – FK, references ATS\_Hierarchy.NodeId.

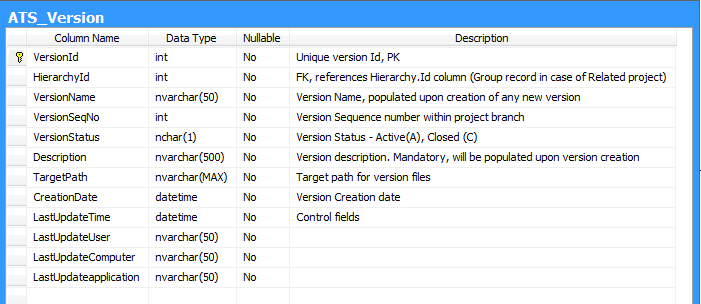
#### ATS\_ProjectStep

ProjectStep is a Reference table, managed by SCD. Project steps definition, a drop down list on project Properties tab.



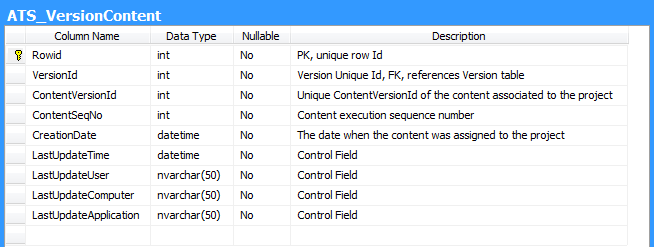
#### ATS\_Version

Versions management.



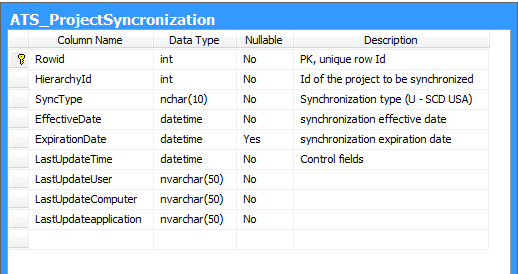
#### ATS\_VersionContent

Content assigned to project version.



#### ATS\_ProjectSynchronization

The table contains projects that will be synchronized with another system.

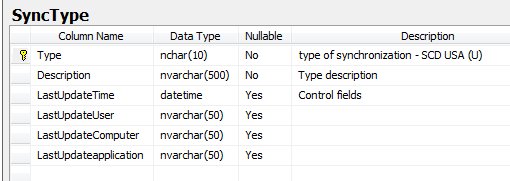


ProjectId – FK, references Project.ProjectId

SyncType – FK, references SynchronizationType.Type

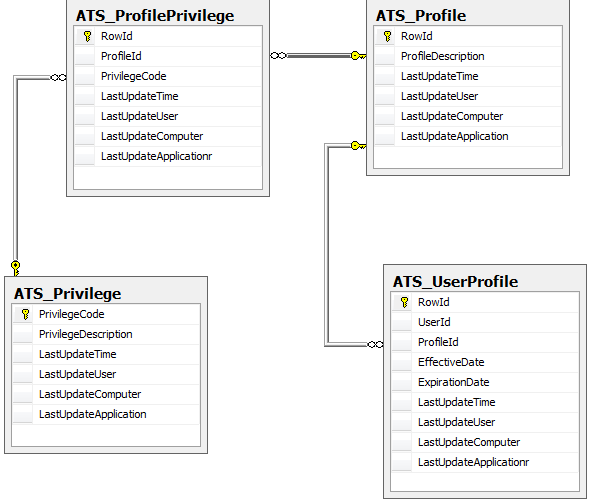
#### ATS\_SynchronizationType

Reference table, describes various synchronization types



## Security

### ERD



### Tables definition

#### ATS\_Privilege

The table lists all system activities that require authorization.

#### ATS\_Profile

The table lists user profiles (roles). In phase 1 the table will include at least 2 basic profiles: Super User and Default User.

#### ATS\_ProfilePrivilege

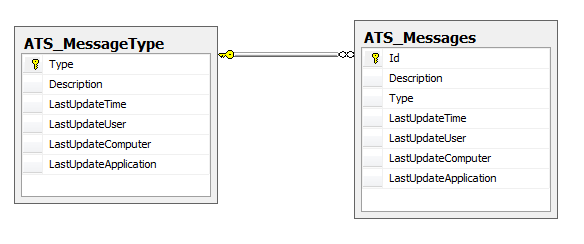
List of activities (PrivilegeCode) that are allowed for the role (ProfileCode).

#### ATS\_UserProfile

The table lists user-profile relationship.

## Messages and Error Handling

### ERD



### Table Definition

#### ATS\_MessageType

Reference table that describes message types: E (error), W (warning), I (information).

#### ATS\_Messages

Message is a Reference table that lists all system messages. MessageDescription is a table driven value (by MessageId); it will be displayed to the user where applicable.

## Operational

System level parameters are stored in ATS\_SystemParameters table.

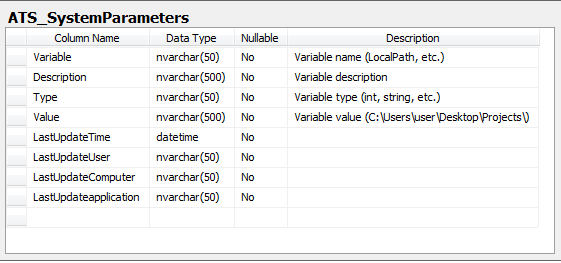
Variables:

ProjectLocalPath

RelatedGroupLocalPath

DefaultProfileId

RequiredDiskSpace



# APIs and Interfaces to Other Modules

The API function will be implement using interfaces, one for each application:

* Content management interface
* Resource management interface
* Project management interface

The interface is an abstract class, which includes a list of method.

The class that will implement the interface must implement all interface functions.

## Content Management

### getTreeObjects function

ContentManagerApiProvider class will implement the interface class of content

Management.

Function prototype:

public List<TreeNode> GetTreeObjects(out Dictionary<int, Folder> folders, out Dictionary<int, Content> contents, out Dictionary<int, ContentVersion> versions)

This function will receive empty parameters of folders, contents and version and return list of tree node.

The returned list is of the content tree that is managed of content management application and there are no allowed actions on this tree. The content tree is read only.

### getContentByContentVersionId function

ContentManagerApiProvider class will implement the interface class of content

management.

public Dictionary<int, ContentVersion> getContentByContentVersionId (List<int> contentVersionsID).

This function will receive list of content version id and will return all the details about this content version.

The returned properties are managed in content management application.

### Resource Management

#### getAllCertificates function

This function will be in resource management interface.

The function prototype:

public Dictionary<int, Certificate> getAllCertificates()

This function will return all the exist certificates (name, id), that will display to the user in order to select appropriate certificate that can assign to any hierarchy on project tree.

The returned properties are managed in resource management application.

#### getCertificatesByWorkstationId

This function will be in resource management interface.

The function prototype:

public Dictionary<int, Certificate> getCertificatesByWorkStationId(string computerName)

This function will return all the certificates details for the specific work station according to computer name.

### Project Management

#### getProjectsByContentVersionId function

This function will be in project management interface.

The function prototype:

public Dictionary<int, Project> getProjectsByContentVersionId(int ContentVersionID)

This function will return all the projects details (name, code, step) that assigned to the delivering content version id.

# Messages

Warning, Information and Error message verbiage is a table driven value and it is retrieved from Message table by MessageId.

Messages:

|  |  |  |
| --- | --- | --- |
| Message ID | Description | Type |
| 100 | The name is already in use. Please specify another name. | E |
| 101 | Are you sure you want to remove the folder? | W |
| 102 | Are you sure you want to disable the note? | W |
| 103 | Are you sure you want to remove the certificate? | W |
| 104 | An object has been updated by another user. Please refresh and try again | E |
| 105 | Unexpected error occurred. Please try again | E |
| 106 | User %name is not authorized to perform this action. | I |
| 107 | Current workstation is not certified for the project.  The following certificates are missing: %list of missing certificates.  Wold youlike to proceed anyway? | I |
| 108 | Two or more Content versions of the same Content have been added. Please remove and try again | E |
| 109 | Two or more certificates with the same Certificate ID have been added. Please remove and try again. | E |
| 110 | Project Code and Step combination is not unique. Please update and try again. | E |
| 111 | Project '%name' is related to the group. Would you like to update all projects belonging to the group? | W |
| 112 | Version %[active versio name] will be closed. Are you sure you want to reopen version %name? | W |
| 114 | Folder '%name' has been created successfully | I |
| 115 | Project '%name' has been created successfully | I |
| 116 | New related project '%name' has been successfully created and assigned to the group | I |
| 117 | Related projects group has been successfully updated. | I |
| 118 | Are you sure you want to create a new version? | W |
| 119 | Content %name1 is referenced by content %name2. Please remove and try again. | E |
| 120 | Content %name is retired. Do you want to assign retired content to the project | W |
| 121 | There is not enough disk space. Please free disk space and try again. | E |
| 122 | Content %name1 and content %name2 are referenced by content %name3. Please remove %name1 or %name2 and try again. | E |
| 123 | Related projects Group Name is already in use. Please specify another name | E |
| 124 | Version Name is already in use. Please specify another name. |  |

# Security

Project Explorer application will receive Username from Access Management application.

List of User privileges will be loaded upon login:

SELECT PrivilegeCode FROM ProfilePrivilege pp

JOIN UserProfile up

ON up.ProfileId=pp.ProfileId

WHERE up.UserId=’&input’

In case when user doesn’t exist in UserProfile table, default profile privileges will be loaded.

Default profile Id will be stored in ATS\_SystemParameters table, ‘DefaultProfileId’ value:

SELECT PrivilegeCode FROM ProfilePrivilege pp

WHERE pp.ProfileId= (select value from ATS\_SystemParameters

where variable=’ DefaultProfileId’

If privilege does not exist in retrieved list then user is not allowed to perform appropriate activity.

The following message will be displayed when user attempts to perform activity that is not associated to user profile:

Message 106:

“User %name is not authorized to perform this action.” (OK).

Menu options and screen buttons security validations will be implemented.

**Privileges list:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Activity** | **User action** | **Implementation** |
| 101 | AddNew | Select folder --> Right click menu --> New | Error 106 |
| 102 | MoveHierarchyBranch | Select folder --> Right click menu --> Cut Or Drag&Drop | Error 106 |
| 103 | DeleteFolder | Select folder --> Right click menu --> Delete Or Keyboard Delete button | Error 106 |
| 104 | ViewDisabledNote | Select folder --> open Notes section | Disabled notes are not displayed |
| 105 | DisableNote | Select note --> Right click --> Disable | Error 106 |
| 106 | UpdateVersion | Select folder --> press Content button | Error 106 |
| 107 | AssignRetiredContent | Drag retired content version from contents tree to project tab --> receive a warning message 119 --> press 'Yes' | Error 106 |
| 108 | AssignCertificate | Select folder --> press Certificate button | Error 106 |
| 109 | RemoveCertificate | Select folder --> select certificate --> right click menu --> Remove | Error 106 |
| 110 | AddNote | Select folder --> Press Add Note button | Error 106 |
| 111 | EditNote | Select Note --> Right click --> Edit | Error 106 |
| 112 | ExecuteContent | Select content --> right click --> Execute or Double click | Error 106 |
| 113 | DisableProject | Select project --> right click --> disable | Error 106 |
| 114 | UpdateDisabledProject | Any action on disabled project | Error 106 |
| 115 | ResumeProject | Select project --> right click --> resume | Error 106 |
| 116 | ExecuteContentClosedVersion | Select content --> right click --> Execute or Double click | Error 106 |
| 118 | ReopenClosedVersion | Select version --> Right Click --> Reopen | Error 106 |
| 119 | CloneProject | Select Project --> Right Click --> Clone --> Regular | Error 106 |
| 120 | CloneRelatedProject | Select Project --> Right Click --> Clone --> Related | Error 106 |
| 124 | UpdateRelatedProject | Press 'Update Group' on message 111 | Error 106 |
| 125 | SplitProjectFromGroup | Press 'Split the Group' on message 111 | Error 106 |
| 126 | ActivateProjectFromUncertifiedWorkstation | Press 'Yes' on message 107 | Error 106 |
| 127 | UpdateTargetDirectory | Press '…' button (Target Directory field). | Error 106 |
| 999 | AllActivities |  |  |

# Assumptions and Open Issues

## Assumptions

N/A

## Open Issues

N/A