Best Practices Working With Multiple CCM Applications.
This document collects best practices to work with Multiple CCM applications in large size enterprise deployment topologies.

Please see Best Practices for CLM Usage Models for general performance considerations. The known expensive scenarios apply as well and should be taken into consideration.

Contents
Best Practices Working With Multiple CCM Applications................................................................. 1
Introducing Multiple CCM Instances.................................................................................................. 2
  Strategies to Distribute Projects – CCM............................................................................................ 2
  Multiple CCM Instances – “Project Move” Scenarios................................................................. 3
Administration ..................................................................................................................................... 4
  Process Sharing ............................................................................................................................. 4
  Project Area Associations ............................................................................................................. 4
Reporting ................................................................................................................................................ 5
Dashboards ........................................................................................................................................... 6
Repository awareness ............................................................................................................................ 8
  Web UI ............................................................................................................................................ 8
  RTC Eclipse client .......................................................................................................................... 8
    Create a descriptive connection name ....................................................................................... 9
Work Items........................................................................................................................................... 10
  Work Item Linking ......................................................................................................................... 11
Planning ................................................................................................................................................ 12
  User Work Environment, Allocation and Scheduled absences..................................................... 13
SCM ..................................................................................................................................................... 13
  Distributed SCM ............................................................................................................................ 13
  SCM Change Set to Work Item Linking ....................................................................................... 15
Locate Change Set................................................................................................................................ 18
SCM Operation Behavior .................................................................................................................... 19
Multi Repository Loading .................................................................................................................... 20
Build ................................................................................................................................................... 20
Introducing Multiple CCM Instances
Sizing should be one of the continuously performed activities when keeping tools in operation. Servers can only maintain certain loads and back up becomes challenging as data size grows.

It is a best practice to plan for the growth potential of your deployment. If the expected amount of concurrent users exceeds the amount that the server can handle, consider using multiple CCM instances. See CLM Sizing Strategy and RDM performance and scalability for details about typical loads.

Strategies to Distribute Projects – CCM
If planning indicates your deployment will likely exceed the load a single server can handle, use these strategies for project distribution across Change and Configuration Management applications.

• Keep project areas that require parent child relationships on one CCM
  o Note that parent child relationships in planning is only supported if both work items are in the same project area the plan resides
  o Consider using cross project planning and tracks relationships instead
• Consider separating project areas which anticipate significant growth into different CCMs
• Consider separating areas based on logical or organizational boundaries
  o Keep users on a single CCM if possible
• Minimize use of distributed SCM where possible
  o Group project areas which share code in the same CCM
• Move project areas with the least possible impact
  o End of a release cycle
Example: planning indicates it is likely that two CCM servers are needed to carry the load for the expected projects.

- Projects 1, 3 and 4 are dependent with respect to work items
- Projects 2 and 5 are dependent with respect to SCM content

Place the two groups of projects on different CCM servers to minimize dependencies between CCM servers.

**Multiple CCM Instances – “Project Move” Scenarios**

If servers reach their limits and become overcommitted it is necessary to take some of the concurrent users and load off. This usually means migrating data. Migrating data is a considerable effort and usually means also a loss of data. RTC does not provide a capability to allow a lossless export and re-import of project areas.

The IBM Integration team provides an RTC Project Area Relocation service which can help migrating project area data such as work items with supporting custom tooling. Please approach IBM for availability and terms of this service.

In general the following approaches are available.

- **Bring over Work Items**
  - CSV Export and Import
    - Bring over open work items, leave completed work behind
    - Keep the old project open for the time being to allow referencing back to it, but limit user capabilities to modify data, eventually make the data read only and abandon it
  - Other Data

- **Bring over SCM Data**
  - Bring over latest baseline from one application repository to another CCM repository using distributed SCM
  - Limit data growth to one CCM
    - Identify self-contained project areas
    - No or limited need to integrate SCM data across repository boundaries
    - Contain SCM data on one CCM server
    - Limit duplicate copies of SCM data in multiple repositories

- **Other Data**
- Build Definitions and build engines have to be recreated manually
- Consider Process Sharing which can be used across servers

It is possible to use the API to add custom automation to further reduce manual work. The Work Item Command Line provides example code that can be re-used.

Administration

Independent of a single CCM application or multiple CCM applications, it is important to understand the Rational Team Concert process fundamentals.

Process Sharing

When setting up projects consider Process Sharing which can be used across servers. Process sharing simplifies managing multiple project areas and their process as well. A project area, potentially on a dedicated server, shares its process. The process is used in other project areas. By preventing local process customization all project areas stay aligned and can advance in parallel. This article provides additional information on process sharing. It is a good practice to test process changes before sharing the new process. Process sharing also provides more capabilities as explained here and even allows Fine-grained Customization of Configuration Data.

Project Area Associations

To be able to link work items and change sets across project areas and across repository boundaries, it is necessary to setup the desired artifact container associations. This has to be done in the project area administration of one of the project areas for each pair of project areas and artifacts.

The images below show the association editor and how to setup associations in the Web UI.

This image shows the same association editor in the RTC Eclipse UI.
The relationship is then shown in the Lifecycle administration page.

It is not necessary to use the Lifecycle project management for this to work. If one of the projects is part of a lifecycle project the related artifacts container will show up. When adding the artifact container to lifecycle projects, the associations have to be set up as well.

**Reporting**

The Jazz Reporting Service Report Builder allows building reports that include data across multiple applications. Applications can be the same applications such as multiple CCM applications or different applications such as QM and RM. The reports can include items located in multiple applications or project areas.

For reporting across servers see the live example ALM 6.0.3 Milestone 4 (work targeted for 6.0.3 Milestone 4, sorted by Product, Component, Status & WI). Note the work items are in multiple CCM applications (jazz03, jazz04).

Reporting usually needs items to belong to a reportable item type such as work items or require relationships between reportable item types to follow.

Relationships between work items are useful to convey certain semantics important especially for reporting purposes. Currently there are only few relationships available to link work items across CCM repositories. It is not possible to create custom link types across CCM repositories. Custom link types are also currently not reportable. See Enhancement 94584 for an enhancement request to mitigate this.
Reporting can create considerable load on the applications. See expensive scenarios and best practices and make sure to understand the implications how to avoid problems.

**Dashboards**
Dashboards can collect and show data across applications. Personal and project dashboards as well as the Mini Dashboard allow selecting the application from which to select the widgets. For an application to be available in the selection it needs to be registered to the same JTS or be a friend of the application showing the dashboard. The image below shows the applications available, including multiple CCM applications that are registered to the same JTS. The different context roots show which one is which.

![Image showing applications available](image.png)

Once an application is selected, specific dashboard widgets for the application are available. Add widgets to show data from this application as desired.

Special widgets show the projects the user is member of. This can be used to simplify overview and navigation for users.
Most of the Dashboard widgets only display information from one application. Report builder allows reporting on data from across multiple applications. Use the Report Builder capability or available lifecycle queries if data across multiple applications needs to be displayed.

Widgets with multiple pages of results place additional demand on the server, while dashboard users are unlikely to click through all of those results. Ensure your dashboard widgets provide a relatively small result set. For dashboards with high or frequent use, avoid including widgets returning a large number of results; alternatively, put them on low use tabs.
Repository awareness

Web UI
If multiple CCM applications are registered to a JTS, the context root of the application is presented in the Web UI for convenience. Navigation to elements is as usual, select the project area from the home menu and work inside the project area.

RTC Eclipse client
In a lot of the regular usage scenarios, it is not relevant to know on which CCM application the project area resides. Like the Web UI, the RTC Eclipse client usually displays context information in the UI as well. It displays the name of the repository connection. A separate repository connection is needed to connect to each CCM server. The image below shows the Eclipse client being used against two CCM servers.
The default connection name is derived from the public URI root. If using a reverse proxy, as suggested, the public URI root is the same for all applications and also for all CCM applications. In the image above the default connection name is ralph@clm.example.com for both connections.

This can cause issues in some dialogs, where it is actually important to know the context. For example, when creating a repository workspace it is necessary to choose the repository to contain the repository workspace and the stream to flow against. This requires selecting the repository based on its connection name.

If using the default connection name in a situation like here, this can cause issues where it is not possible to understand which repository is selected as can be seen below.

Create a descriptive connection name
The best way to mitigate this is to manually change the connection name when creating the repository connection or using the properties dialog. For example, add the context root to the name.
Now it is easy to understand the context.

![Select a Stream](image1.png)

The connection name is also shown in other views. The information is most important for distributed SCM when deciding on the repository to create or select items.

![Repository Connections](image2.png)

**Work Items**

There are no special best practices just for work items. Work items are created from within the project area. If work items are created in the context of a link operation the project areas available to look for work items are determined by:

- The link type
- The project area the source element is owned by
- The project area associations this project area has
- The repository this project area is located in
**Work Item Linking**

RTC provides different link types that can be used between work items and other artifacts. The different link types can be categorized into two different flavors.

**Work Item links** are only local and can be only created within one CCM application but across project area boundaries. Examples are:

- Parent/child
- Duplicate
- Related Work Item

**CLM links** can be created across application boundaries. Examples are:

- Tracks->Contributes To
- Affected By Defect->Affects Plan Item
- Related Change Request->Related Change Request

CLM links are also sometimes referred to as OSLC links.

Consider which link types between which elements should be used and consider implementing process advisors that prevent creating link types that should not be used where possible. This is especially important if a clean link schema is desired in a multiple application setup where some links work across repositories and others don’t.

This can be achieved with custom work item save and other advisors.

Due to Enhancement 228421: Adding links of type "Tracks", "Contributes To" and "Related Change Request" does not trigger a work item save participant it is only possible to detect the creation of work item links. It is not possible to create advisors that react on saving Work Item CLM links.

Gap: there is a need to have more link types that work across CCM repositories and have a consistent behavior: Enhancement 394584: Provide more cross repository work item link types to support richer tracking in multiple repository environments.
Planning
When using cross project planning consider a hierarchical approach.

On the lowest level, use project areas and team areas to structure and organize users and plan their work, e.g. plan the work that needs to be done by the team. At this level it is possible to use the agile planning, complexity and the parent/child relationship. Effort at this level is ideally aggregated using plan items. Note that the order of execution of the work items associated to a user is determined based on the order in the “My work” view. It is possible to reorder the order in the road map view, but in general RTC supports bottom up planning and not top down planning. Plan snapshots can be created to make the current planned times available to tracking cross project plans in other project areas.

Cross project plans collect and show plan information using the tracks relationship between work items. Cross project plans work across project areas and repository boundaries. Work items that track effort are located in the cross project plan. These work items can track plan (and execution) items in other project areas by creating a tracks relationship to such a work item.

If a tracks relationship is created to a plan item that organizes several tasks e.g. using parent/child relationship, it is important to note that the cross project plan only shows the aggregated effort for the plan item. It does not show the effort for each child item below.

Hierarchical plan display only works for one link type. The view will not show other relationships in addition. If the hierarchy is for the Tracks->Contributes to link, Parent->Child relationships will not be displayed in this view.

Drag and Drop is available in plans and can be used to create relationships between work items. This is available if the view displays a tree. The drag destination is the work item id and summary. It is not possible to drop on a link column or work items that are out of plan. The link that is created is the link type that is used to create the tree hierarchy.

It is possible to use Drag and Drop between two plan views on a link column to create relationships. This currently only works if the plans are within the same project area. The relationship that is created is determined by the link type that is selected for the column.

Ideally drag and drop would be supported for more scenarios.

There is an issue with drag and drop in plans documented in 393598: Drag&Drop work items on reference targets to create relationships fails if the drag work item lives in a different CCM repository and corrupts the plan. The scenario is not supported, but the feature does not prevent it from happening.

For optimal performance please be aware of and follow the plan loading best practices.
**User Work Environment, Allocation and Scheduled absences**

Scheduled Absences, allocation and work environment for all users are maintained within each CCM Server. Users that work against multiple CCMs will be scheduled as 100% available for each CCM. This can lead to inconsistent data such as overbooking users and different scheduled absences.

Consider a departmental approach if possible, where users contribute planned and measured work only to one CCM.

If a departmental approach is not possible, consider API based automation to consistently update scheduled absences and user allocation.

It is possible to use external and internal APIs to add automation to handle the user data and keep it consistent. Here are some examples that use this API:

The [JAM – Jazz Workallocation Manager](https://www.ibm.com) A lightweight Web client to manage the Workallocation of users or team within Jazz/RTC has been created by a customer to simplify this task.

Some more API examples can be found in the blog post [Manage Scheduled Absences Using The PlainJava Client Libraries](https://www.ibm.com) and there are more API examples available that are currently not yet published.

Note: This likely requires use of internal API. Consider requesting the API be published.

**SCM**

**Distributed SCM**

Distributed SCM must be enabled on all servers that should participate.
In addition, users that need to use distributed SCM require a special permission in each involved project area. Provide a special role in each project area that has the permission to replicate change sets and identify users that should have this role.

Technical users such as build users need this role to be able to work across repository boundaries.

Distributed SCM and normal SCM are not that different. The basic model is to have a repository workspace and a stream. The repository workspace is the owner’s sandbox and the stream is the coordinating entity for multiple users. Change sets flow between the repository workspace and the stream. The repository workspace and stream can be in different repositories in which case distributed SCM adds the capabilities to deliver the changes across server boundaries. Delivery of a change set across repository boundaries fundamentally means replicating the change set from one repository to the other repository.

Make sure users that work across repository boundaries have a role that permits replicating change sets.

As mentioned above, rename the repository connection names to get better location information for creating repository workspaces and streams.
Distributed SCM only replicates the current baseline. It is possible to manually accept baselines that are needed and then deliver them across to another repository.

New changes in one repository are not automatically visible in the other remote repository. Snapshots are local to a repository. They cannot be replicated across multiple CCM repositories and selecting a snapshot owner (stream) in a different repository is not possible.

Consider automation to synchronize baselines and recreate snapshots from baselines.

It requires careful planning to split the users across CCM servers to balance load. It is therefore important to make sure that relevant load is actually moved to another server and it is clear who is working against which repository. This is especially true for SCM operations which usually generate more load than using work items.

**SCM Change Set to Work Item Linking**

An important benefit of using RTC is due to being able to organize work and code changes with work items. The default behavior in the RTC Eclipse client can be used against repository workspaces loaded from different repositories. Use the current work item feature and drag and drop to associate work items with SCVM change sets.

To set a work item as current, use the context menu on the work item in the RTC Eclipse client.
The action in the status bar will show the current work item.

If change sets are created while the current work item is set, the current work item will be associated with the change set. It is also possible to drag and drop a work item on to the current work item status bar to make it the current work.
The current work status bar is shown by default and can be configured in the work item preferences here, if it was accidentally switched off.

Use drag and drop onto the pending changes view to drop work items on change sets. This is the easiest way besides the current work to link a work item to a change set. The relationship is created regardless of if a work item and change set are located in the same repository or in a different repository.

The change set link on the work item will show if the change set and the work item are in different repositories. Change sets in a different repository show up as “Change Sets (remote)” link on the work item.

The third way to link work items to change sets is to use Associate Work Item... or to use Associate Change Request... from the context menu of the change set.

Which of the two choices to use, depends on the relative location of the work item and the change set.
• Associate Work Item..., only allows selecting and linking work items from the same repository; only project areas in the repository the change set is located can be chosen to search for work items
• Associate Change Request..., only allows selecting and linking to work items that are owned by project areas that are associated to the project area that owns the change set using the Provides Change Sets association.

The repository workspace is owned by the user and is not owned by a project area. Given this, how does “Associate Change Request” determine which project area the change set belongs to in order to provide the associated project areas to select the work item? The relationship is created by the component. Any change that is checked into a repository workspace is checked into a component. The owner of this component is taken as the project area context of the change set.

**Locate Change Set**

Locate change set can be used to identify if change sets are available in streams and repository workspaces. Change sets can be added directly using the context menu or drag and drop. An alternative approach is to provide work items as input. Locate change Set searches the work items, including their child work item hierarchies, to find associated change sets and provides the change sets found for selection input. Locate change set supports drag and drop of change sets from arbitrary repositories. It also allows choosing work items from all available project areas and adding them to the Change Sets selection.

Locate Change Set is able to locate the change sets that are selected in the streams and repository workspaces of the repository they live in. It is possible to add search targets from any repository.

The change sets section only uses the “Change Set” relationship on the work item to find change sets.

See work item 2 in CCM 1 has 3 change sets. One in another repository

![Image of links](image-url)
Only the ones in the local repository show up as change sets.

This behavior is surprising and confusing because the remote change set links are presented in several other views. It is not necessarily wrong, but confusing.

**SCM Operation Behavior**

SCM related operation behavior (advisors/pre-conditions) that is built into the product only provide limited functionality or only work in the **local repository containing the change set**. Impacted advisors are:

- Required Work Items and Comments
- Required Work Item Approval
- Required Work Items to Match Query
- Restrict Change set delivery to Components

If the change set and the work item are not in the same repository, the advisor “Required Work Items and Comments” only works in this configuration:

This allows making sure that a change set must have a work item or a change request linked to it.

It does not allow making sure that a remote work item (considered a Change Request) is approved, planned for the current iteration and other constraints.
See enhancement request: [396190: Allow the Source Control Deliver (server) Preconditions "Require Work Items and Comments" and "Require Work Item Approval" to work with associated change requests (OSLC Work Item Link Type)] for additional functionality. The other preconditions don’t work for associated Change Requests.

**Multi Repository Loading**

It is not strictly required to use distributed SCM to work with data that is stored in different repositories. RTC SCM can be used to load repository workspaces from different servers to the local disk. The load will always be possible, as long as there is no overlap in the file structure. Once the content is loaded to disk the user can develop. The Jazz SCM tracks the loaded files and allows editing and checking in the changes into the repository workspace in its related CLM repository.

As long as the repository workspace and its flow target are in the same repository there is no distributed SCM involved during deliver and accept operations. This is just the normal way of operation.

Limitations in this scenario are the lack of automation to configure or reproduce this reliable in the tool. There is no concept that allows to create composite repository workspaces or to create composite snapshots across CCM applications. Each CCM application only knows about the elements that are located in it.

**Build**

The Build related artifacts are owned by a project area in a CCM application.

Build engines can only be connected to one repository so it is necessary to have enough build engines available. Note: the RTC/Jenkins integration seems to be more flexible as it is possible to create multiple Jenkins build engines that connect to the same Jenkins Server and Jenkins builds that work with those build engines.

Build engines can be used across project areas in a repository. Make sure to have enough bandwidth to be able to serve all the build requests.

Build engines load data from the SCM system. To keep the SCM access load as small as possible use content caching proxies between the build server farms and the CCM application.

It is not possible to copy build definitions. It is possible to create a build definition from an existing build definition. This is currently only supported within a CCM repository.

It is possible to run a build against a repository workspace that flows to a stream in a different repository. Limitations apply to some of the built in JBE build options such as post build deliver that only work within a CCM repository.

Snapshots referred to in the build result are local to the repository containing the build result.
It is possible to load multiple repository workspaces from multiple CCM repositories for a build. This would require using the RTC SCM Command Line. Creating snapshots and baselines would also require the RTC SCM Command Line. It is possible to load the full content directly from the build due to the multiple Snapshots.

Work Items can be created against build results. These work items can only reside in the project area that owns the build definition.

**General Best Practices**

**Project Distribution**
If possible, identify project areas that are needed and organize them by dependencies between the project areas. Look for dependencies based on work item relationships as well as shared or dependent code in source control.

If at all possible, deploy dependent project areas on the same CCM. This allows using all capabilities available within a CCM repository.

**External References**
RTC work items have a unique ID, however these IDs are only unique within the RTC repository they live in. If referencing work items in documents or spreadsheets, it is a good practice to store information in addition to the work item ID to be able to uniquely identify the work item. It is possible to use the context root or project area name. It is also possible to copy the ID and the summary of the work item including the work item link and store this information. It is possible to use a calculated value and a unique prefix to calculate a custom unique ID as well.

The RTC SCM component provides the capability to provide unique names for components and other objects. This is only possible within one repository. It is possible to create automation to detect name collisions and to potentially resolve these issues.

Please note RTC manages unique identifiers that are unique regardless of the human readable name.

**Related Work Items**

399219: Enhancement: Explore simplification of cross repository SCM operations
398903: Enhancement: Explore simplification of cross repository work item linking and planning
Additional Reading

Multiple CCM Applications - A User Perspective

Planning for multiple Jazz server instances

Best Practices for CLM Usage Models