In IBM® Rational Team Concert™, you have access to the feature-rich IBM® Rational® Jazz™ source control management (SCM) system that is subtly different from other such systems that are available in the marketplace. This article provides an overview of the various components that comprise the SCM system and offers both basic and detailed operational overviews on how you can use it.

While IBM® Rational ®Jazz™ version control shares similar names to other version control systems, subtle differences do exist. First, we will explore the components that comprise Jazz source control management (SCM). Second, we will look at how changes are calculated and moved between Jazz SCM components. The following diagram shows an overview of all elements;

**Figure 1. Jazz source control management (SCM) diagram**

**Components of Jazz SCM**

The component composition of Jazz SCM is subtly different to many available SCM systems. Before we continue it is important to have common understanding of what these components are and their intended purpose.
Change set
A change set is a collection of file or directory changes that are grouped together to form a single unit. It is this complete unit that is committed to a single component and associated with work item(s). Change sets are created when you check-in changes from your local workspace into a repository workspace.

Component
This is the final destination for all change sets and baselines. When using either Eclipse or Microsoft® Visual Studio, files and directories are shared into a component at the project or solution level, respectively.

Stream
Use streams to create isolated configurations of change sets and baselines through which users can deliver their change sets to a component. Depending upon the process configuration of the IBM® Rational Team Concert™ project, all users can create streams or a restricted subset.

As new change sets are delivered, the stream configuration updates to reflect its evolution. By comparing the differences between the stream and the repository workspace configuration rules, the system identifies change sets that users must accept into their repository workspace to maintain synchronization with the stream.

A stream is also a container for snapshot information.

Repository workspace
A repository workspace is a storage area on the Rational Team Concert server. The workspace contains a configuration consisting of components and baselines, storage for holding undelivered change sets and baselines (not yet stored within a component), and storage for snapshots not yet promoted to a stream. Change sets that are in a repository workspace can be delivered to a corresponding component.

Local workspace
This is a directory on the user's local workstation where they have loaded a repository workspace. Any changes that users make in their local workspaces are identified and highlighted as being candidates for being checked into the loaded repository workspace where they are grouped within a change set.

Baseline
When a repository workspace represents a configuration of file or directory versions that users want to share with others or mark for future use, then a baseline must be created. These markers are delivered to the component in the same manner that change sets are. A baseline can only be applied to change sets that reside within a single component.
Flow target

Both repository workspaces and streams can have a flow target relationship that is defined to another repository workspace or stream. Each repository workspace or stream can have many of these flow targets defined, but might only have a single one active. It is the active flow target that allows incoming and outgoing changes to be calculated.

How it works - Basic operation

If you use the web-based version of Rational Team Concert, then you can use Jazz source control management (SCM) functionality directly through a stream, which is a great way for users such as technical authors to place their assets under configuration control. They can also contribute to and collaborate with the broader development team.

For developers who need more configuration management features, they can use the Rational Team Concert Eclipse IDE, Microsoft Visual Studio Rational Team Concert plug-in, or the feature-rich Jazz SCM command line interface. The rest of this article assumes that you are using one of these interfaces.

To use Jazz SCM, the first thing you must do is create a component that does not reference any other components. Then, users must create a repository workspace based on the stream that they want to work with. Lastly, users must load the repository workspace into a local workspace (such as an Eclipse workspace).

After the local workspace is set up users can then share Eclipse projects or Microsoft Visual Studio solutions that are in the local workspace with the repository workspace. They can specify either an existing component or a new one into which its assets will reside. Within the repository workspace the shared assets are organized into a new change set.

Within a repository workspace, users can suspend, resume (if previously suspended), or deliver change sets through the stream and to a component. Each change set can only reside within a single component. The destination component is determined by the assets that are in it. For example, the sharing operation determines the top-level directory assets to be associated with a specific component.

Any modifications that are made within a local workspace to either shared Eclipse projects or MS Visual Studio solutions are identified as items that can be checked in to the repository workspace. At check-in, these items are packaged within one or more change sets and must be delivered through the stream to a component.
**Figure 2. Jazz SCM core operations**

![Diagram of Jazz SCM core operations]

**Share**
Taking an existing Eclipse project or Visual Studio solution and sharing it to Jazz SCM for configuration management purposes.

**Checking in changes**
When users check in their changes, they essentially package changes from a local workspace and copy them to a change set that is stored in a repository workspace.

**Deliver**
When you deliver a change set, you commit it from a repository workspace to a component.

**Promote**
The act of promoting means that you are moving the storage location of a snapshot from where it resides to another repository workspace or stream.

**Associate**
When you associate change sets, you create a relationship between the change sets and either Rational Team Concert work items or OSLC change sets.

**Lock**
This operation locks a specified asset within the context of the stream, which prevents other users from modifying the stream. The user that initiated the lock operation of that stream can still modify it.

**Unlock**
This operation restores the ability of users to modify a locked asset.

**Suspend**
This operation is used on a change set that is stored in a repository workspace. The repository workspace configuration is modified so that it does not include the change set. The local workspace that is associated with the repository workspace is updated immediately to ensure the change set is not included. The change set is still stored within the repository workspace and can be resumed at any time.
Resume

This operation is used on a suspended change set that is in a repository workspace. The repository workspace configuration is updated to include the resumed change set. At this point, the local workspace might have to accept the change set and perform a merge with any changes made since it was suspended.

How it works – Detailed Examination

In this section we will explore in more detail the mechanics of how Jazz source control management (SCM) works in the context of a new project. The following aspects are explained:

• What is provided in a default Jazz SCM Rational Team Concert project
• How initial assets are added to Jazz SCM
• How to record a component configuration for future reference and sharing purposes
• How to work collaboratively on shared code
• How to record a stream configuration for future replication
• How to create a new stream and deviate between streams

By the end of this section you will have an appreciation of the progression of assets as they are added to and modified within SCM control, how to uniquely mark a release and how you can do maintenance without clashing with work for the next release.

Create a project in Rational Team Concert

When you create a new project, the default behavior (dependent upon the process template being used) is for both a default component and stream to be automatically created. The stream is configured so that it is associated with the default component that is using its initial baseline. This is the starting configuration of the stream.

The following figure shows the structure of components after you create a project and a user’s repository workspace that is associated with the stream. At this point no files or directories have been committed to Jazz source control.
Share project artifacts with Jazz source control

Depending on which IDE you use, it is necessary to create a project (or solution) and share this with Jazz source control. To share, you must specify a target component. After you share, the assets are grouped together into a change set and stored within the repository workspace. For these assets to be visible and usable by other Jazz source control users, you must deliver the change set to the component that you specified when you shared.

Each change set is assigned a unique number. Whilst the number is not visible through the IDE clients, it is visible to a user through the Jazz source control command line interface (CLI). See Figure 4.

Figure 4. Jazz source control CLI displays change set number

When the change set is delivered, the stream configuration is updated to reflect that its current status is the combination of the starting baseline plus the new change set. These actions are shown in Figure 5.

1. The share operation packages up the selected project/solution into a new change-set and moves it to a designated repository workspace.
2. Deliver moves the change-set to the component specified at the time of the share operation.
Create the baseline for a component

Now that your project is shared, it is accessible by all repository workspaces with an active flow target to the stream through which project was delivered. Other streams cannot see this project because it is contained within a change set that is outside of the initial baseline.

Configurations for new streams and repository workspaces are comprised of a variable number of paired component and baseline. Whilst it is possible to manually reproduce the desired configuration by loading specific change sets such as #1 into a repository workspace and delivering it, the preferred option is to create a new component baseline that encapsulates the new change set and its shared project.

Users can create baselines within a repository workspace, but the baseline must be delivered to the appropriate component. Figure 6 illustrates the following actions:

1. The create baseline operation is performed on a component within a repository workspace. The baseline object is a marker of the configuration rule for the component. Upon creation, the configuration rule of the repository workspace is updated to reflect the new baseline. In this example, the rule has been modified to reference BL1 for ComponentA. The baseline is temporarily stored within the repository workspace.

2. The deliver operation moves the baseline to the component where it can be used within other stream or repository workspace configuration rules. During this operation the stream through which it was delivered updates its configuration rule to reflect the baseline, in this example this is BL1 for ComponentA:
Baseline 1 (BL1) is created within a repository workspace whose configuration rule updates to reflect the new baseline. The baseline is now located within the repository workspace awaiting delivery to the component. After the baseline is delivered, it moves into the component and the stream configuration is updated to reflect the new baseline. The baseline can now be referenced within other stream or repository workspace configurations.

A baseline has the scope of a single component. If a configuration needs to be marked that spans multiple components a snapshot needs to be created, this is covered later.

**Working on shared code**

After being delivered to a component, assets can evolve in a controlled manner. Changes that are made to assets are required to be checked-in to the repository workspace. By default users must initiate this action, but it is possible with the Eclipse IDE to have the check-in operation automatically executed when files are saved.

Modification(s) made to a shared asset are identified as a candidate for check-in. Then a new change set is created within the repository workspace. Subsequent check-in operations might associate changes to this change set, or the user can explicitly request that a new change set is created.

With multiple change sets in the repository workspace, users can choose which change set to associate changes with, or allow the IDE client to select one. If users select the change set, they must consider the following constraint: Once an asset has been modified and added to a change set any subsequent changes to that same asset must be amended to the original change set. When the change set has been delivered and no longer resides within the repository workspace, this constraint is reset.
Association of change sets with an Rational Team Concert record (or external OSLC change request) is an optional, yet highly recommended step that increases auditability. It also aids in constructing software based upon specific changes identified through their associated Rational Team Concert work item or OSLC change request record. This practice can be enforced through Rational Team Concert process customization capabilities.

The final step is to deliver the change sets to the component via the stream that is identified as the active flow target. The following actions are illustrated in Figure 7:

1. The check-in operation packages the modified files and directories from the local workspace, moves them to the repository workspace from which the project was loaded and places them within a change set. The configuration rule of the repository workspace updates to reflect that it is now using the combination of baseline BL1 plus change-set #2 for ComponentA.
2. (Optional): The associate operation allows a user to create an association between a change set and either an Rational Team Concert work item or an OSLC change request.
3. The deliver operation moves the change set to the component. During this operation the stream through which it was delivered updates its configuration rule to reflect the baseline, in this example it now uses the combination of baseline BL1 plus change-set #2 for ComponentA.

**Figure 7. Working on shared code, deliver operation**

Newly delivered change sets are located within the component and reside outside of any existing baselines and relate through the stream and repository workspace configurations. As more file and directory changes are made, stream configurations evolve, and it becomes necessary to take snapshots and store these configurations for various reasons, such as keeping records of new configurations, sharing configurations between streams and for initial stream configuration.
Multiple users and their repository workspaces can use the same stream for collaboration purposes. Delivered change sets are highlighted to each repository workspace attached to the stream as candidates for accepting into the repository workspace.

Figure 8 shows the sequence of the following actions needed to accept changes that are delivered by another repository workspace:

1. Following the delivery of a change set, repository workspaces with an active flow target to the stream through which it was delivered detect that their configuration is different from the stream and identify the change sets as needing to be accepted into the repository workspace.
2. (Optional) If the repository workspace is loaded to a local workspace then the content of the accepted change set is automatically loaded into it.

In this example, accepting change set #2 results in the repository workspace configuration rule for ComponentA being updated to represent BL1 plus change set #2. The content of this change set, file HelloWorld.h, is automatically loaded into the local workspace. Note that when you view the configuration rule of a stream or repository workspace only the baseline is shown.
Create and store a snapshot

Figure 9 depicts how development has progressed for the shared assets we have been following. A second component has been created and some new assets have been shared to it. The stream and repository workspace configurations have been updated to reflect this:

- ComponentA is being picked up as baseline BL1 plus change set #2.
- ComponentB is being picked up as baseline BL0.1
Figure 9. Create a snapshot - Configuration before

The team working on the stream believes that its configuration represents its first release and wants to record it for audit and recreation purposes. To achieve this goal, a new snapshot must be created.

You can create a snapshot either on a repository workspace configuration or directly on a stream. We will look at what happens for both of these variations. Each shares the same eventual result.

Note that it is a feasible strategy to define a stream whose sole purpose is to be a storage container for snapshots. For example, consider a release stream into which all release related snapshots are promoted. This stream would never be actively used other than for locating snapshots for purposes such as reconstructing a release (create repository workspace from a snapshots configuration) or creating streams for purposes such as maintenance work.

Create a snapshot in a repository workspace

In Figure 10, we illustrate the following actions that take place when you create a snapshot within a repository workspace:
1. Creation of a snapshot is requested to occur within the repository workspace. Two activities can take place: possible creation of a new baseline and a snapshot.
   a. For each configuration rule that includes baseline plus change sets, a new baseline is created and placed within the repository workspace for delivery to the appropriate component. The configuration rules of the repository workspace change to reflect any newly created baselines. In this example, ComponentA is now reflecting baseline Release_1.0, which is also the name of the newly requested snapshot.
   b. A new snapshot is created and stored within the repository workspace. A snapshot contains a replica of the repository workspace's configuration rules. This allows the snapshot to be used to recreate this configuration or to represent a starting configuration for a new stream.

2. Newly created baselines must be delivered to the component. In this example, baseline Release_1.0 is being delivered to ComponentA.

3. (Optional) Because snapshots are the mechanism to record configurations that span multiple components, they are extremely important and must be located in an area where they can be easily located. It is for this reason that keeping a snapshot within a repository workspace may not be a good, long-term, arrangement. The promote operation allows for a snapshot to be relocated from a repository workspace to a stream. In this example, snapshot Release_1.0 is promoted from the repository workspace to the stream.

Create a snapshot directly on a stream

In addition to creating a snapshot within a repository workspace it is possible to create one directly within a stream, which is illustrated in the following figure:
1. A request is made to create a snapshot to occur on a stream. Two automatic operations occur as a result: The creation of any required baselines, and the creation of the snapshot.
   a. For each configuration rule that includes baseline plus change sets, a new baseline is created and placed directly within the appropriate component. The configuration rules of the stream change to reflect any newly created baselines. In this example, ComponentA is now reflecting baseline Release_1.0, which is also the name of the newly requested snapshot.
   b. A new snapshot is created and stored within the stream. A snapshot contains a replica of the repository workspaces configuration rules. This allows the snapshot to be used to recreate this configuration or to represent a starting configuration for a new stream.

2. (Optional): If new baselines were created then the change in the stream configuration rules will cause any repository workspaces with active flow targets to the stream to identify the new baseline as something that they should accept to maintain synchronization with the stream.

**Stream deviation and change flow between streams**

Continuing our example, the development team has just made their first release and recorded its configuration in the snapshot “Release_1.0”. They decide to continue using the existing stream for work on the next release, a new stream is required for isolating any maintenance work for the current release.

By using the snapshot Release_1.0, a new stream is created named MaintStream whose initial configuration rules mirror those defined in the snapshot. Any activities that relate to the maintenance of Release 1.0 can now be performed within a repository workspace that is based upon this stream and which has this new stream as its active flow target.
Sometimes an essential bug fix that is conducted through maintenance activities must be shared with the ongoing development activities. To coordinate these activities, consider naming someone to the role of 'integrator'. Ensure that this resource understands the code base and is therefore able to perform educated merges (if necessary). The procedure for performing this scenario is comprised of three distinct steps.

**Step one: Create a repository workspace to be used to control the flow and integration of code**

In Figure 12, the three steps to create an integration repository workspace are outlined.

1. Create a fresh repository workspace that has an active flow target with stream DevStream.
2. Load the repository workspace (preferably into a clean local workspace).
3. Ensure that the repository workspace reflects the current latest configuration of stream DevStream by accepting any outstanding changes.

**Figure 12. Configuration of a new repository workspace**

**Step two: Obtain the necessary changes from MaintStream and integrate them with the latest DevStream code**

In figure 13, we show five actions that must happen to consume changes from MaintStream into the integration repository workspace:

1. Create a new flow target from the repository workspace to stream MaintStream from which bug fixes are to be obtained.
2. Make this new flow target the active one.
3. The repository workspace will identify all bug fixes that reside within stream MaintStream but which are not present in the repository workspace, these are also the changes that are not included within stream DevStream configuration.
4. Accept all bug fixes that are to be integrated into DevStream stream into the repository workspace resolving any change set dependencies.
5. Ensure that the repository workspace contents can be built successfully and that the resulting software passes all necessary tests.
Figure 13. Accept changes from stream containing desired changes

Step three: Deliver the integrated changes to DevStream

The final steps to take deliver the changes that now reside within the integration repository workspace into the DevStream, these steps are illustrated in Figure 14.

1. Reactivate the flow target from the repository workspace to stream DevStream.
2. The repository workspace will now identify outgoing changes that reside within the repository workspace but which are not part of the development stream.
3. Deliver all changes to stream DevStream.

Figure 14. Deliver integrated changes
Related topics

- Find **Rational Team Concert** articles and links to many other resources on the IBM developerWorks page. You can also watch the **Using Rational Team Concert in a globally distributed team** webcast or a demonstration of the **Dashboards and reports**, or listen to the podcast about **IBM Rational Team Concert and Jazz**.

- Visit the **Rational software area on developerWorks** for technical resources and best practices for Rational Software Delivery Platform products.
  - Attend a **free developerWorks Live! briefing** to get up-to-speed quickly on IBM products and tools, as well as IT industry trends.
  - Watch **developerWorks on-demand demos**, ranging from product installation and setup demos for beginners to advanced functionality for experienced developers.

- Improve your skills. Check the **Rational training and certification** catalog, which includes many types of courses on a wide range of topics. You can take some of them anywhere, any time, and many of the “Getting Started” ones are free.

- Rational Team Concert trial downloads (free):
  - **Enterprise Edition**
  - **Express Edition**
  - **Standard Edition**