StarTeam 16.1 Update 1
Synchronizer for Quality Center User Guide

Borland®
(A MICRO FOCUS COMPANY)

MICRO FOCUS®
Leading the Evolution™

Contents

Preface

This manual explains how to synchronize the following defect-tracking applications:

- StarTeam
- Quality Center®

StarTeam Synchronizer for Quality Center, also referred to as *the Synchronizer*, enables you to record defects that Quality Center generates as StarTeam change requests and vice versa.

Related Topics

Support on page 5

Support

Borland (a Micro Focus company) is committed to providing world-class services in the areas of consulting and technical support. Qualified technical support engineers are prepared to handle your support needs on a case-by-case basis or in an ongoing partnership. Micro Focus provides worldwide support, delivering timely, reliable service to ensure every customer's business success.

For more information about Micro Focus support services, visit *http://supportline.microfocus.com*, the Micro Focus Supportline Web site, where registered users can find product upgrades, product documentation, product support as well as previous versions of a Micro Focus product

When contacting support, be prepared to provide complete information about your environment, the product version, and a detailed description of the problem.

For support on third-party tools or documentation, contact the vendor of the tool.

Related Topics

Preface on page 5

Using Quality Center Synchronizer

StarTeam Synchronizer for Quality Center ensures that the same data appears in both Quality Center and the database that the StarTeam Server uses. This synchronization provides access to the latest information about defects, whether Quality Center or StarTeam processes them. Quality Center can add defects, StarTeam can indicate whether the defects have been fixed, and vice versa. Team members do not need to be aware of where the defect was last processed. The latest data is available at all times, as long as the databases are synchronized frequently.

StarTeam Synchronizer for Quality Center uses ODBC drivers to access the Quality Center SQL-compliant relational database. It relies on the StarTeam SDK to access the StarTeam Server, which accesses the StarTeam database. The Synchronizer transfers data about Quality Center defects to the StarTeam database and transfers information about StarTeam defects, called *change requests*, to the Quality Center database.

The Synchronizer uses the Open Test Architecture (OTA), Windows-only APIs to write to the Quality Center database and to read some data. However, the Synchronizer still reads most of the Quality Center data through direct database calls.

The Synchronizer must be run on a Windows operating system because of the OTA APIs.

Related Topics

Installation Overview on page 7
Synchronizing the Applications on page 8
Custom Fields on page 10

Installation Overview

Before installing the Synchronizer, make sure your computer meets the requirements described in *System Requirements* on page 6.

To install the Synchronizer, run the file `ST_QC_Synchronizer.exe` and follow the onscreen directions. `ST_QC_Synchronizer.exe` is a self-extracting executable that, by default, creates a folder named `StarTeam QualityCenter Synchronizer` in your Temp folder. It also adds files like `setup.exe`, which automatically installs the integration files and runs the self-extracting executable `StarTeam-runtime.exe`. `StarTeam-runtime.exe` installs the StarTeam SDK, which enables the integration to access the StarTeam Server.

If you are using the StarTeam Server for Windows and installing over a previous version of the Synchronizer, the installer renames the following files to preserve any changes you might have made to them:

- `BugSync.ini` becomes `BugSync.000`
- `run.bat` becomes `run.000`
- `run-again.bat` becomes `run-again.000`

Synchronizing the Applications

The process of synchronizing the Quality Center and StarTeam databases requires adjustments and customizations to both databases.

To synchronize the two applications effectively:

1. Identify the data specific to each defect and change request that both applications must access.

   This data must be stored in and synchronized between both databases. Identify the fields that store this data as well as the values that can or must be stored in them.

2. Compare the fields from one application with those in the other.

   Use the Synchronizer properties file `BugSync.ini` to designate the fields in each application that will store the same data. Generally, the mapping of a field from each application to create a pair of related fields is sufficient. However, you can map the same Quality Center field to multiple StarTeam fields and vice versa. For mapping recommendations and suggestions, see *Appendix B - Quality Center Defect Fields* on page 53 and *Appendix A - StarTeam Change Request Fields* on page 39.

3. To store data that cannot be mapped directly from one existing field to another, create custom fields in both databases.

   For more information, see *Custom Fields* on page 21

4. Create a synchronization field for each application that acts as a flag and controls which defects are transferred to the other application's database.

   📝 **Note:** You might not want some defects transferred from one database to the other.

   The initial `BugSync.ini` file uses the custom field `BG_USER_02` as the flag field in Quality Center and the custom field `Usr_Sync` as the flag field in StarTeam. However, you can use any two custom fields of appropriate types.

   In Quality Center, you can create a string field or a custom list for the synchronization field. The custom list must contain the values **Yes** and **No**. In StarTeam, the custom field must be an enumerated field or a text field with the values **Yes** and **No**. The Synchronizer does not require **Yes** and **No** to be case-sensitive. You can specify either **Yes** or **No** as the default value.

   If a defect contains a value of **Yes** in this field, the Synchronizer either creates a defect or updates an existing defect in the other application's database. If a defect contains a value of **No** in this field or if the field is blank, the Synchronizer neither adds it to the other application's database nor, if it already exists there, modifies it.

   For more information, see *Custom Fields* on page 21.

5. Create a properties file that indicates which fields of information are transferred and in which directions, as follows:

   a) Change the `vts_parameters` and `mtd_ parameters` to indicate the source of data for each application.
   b) Review the `mtd_project` mapping, which contains a valid value for the `Project (BG_PROJECT)` field.

Alternatively, you can comment out this directive, in which case all defects in the Quality Center database will be synchronized.

c) Map other fields as appropriate.
For more information, see *Quality Center Fields That Contain HTML* on page 26.

> **Note:** Use the initial properties file, `BugSync.ini`, as a template and modify a copy of the file. This approach allows you to review the original settings of `BugSync.ini` when you debug your own properties file. For more information, see the initial properties file, `BugSync.ini`, in *Appendix C - Initial Synchronizer Properties File* on page 60.

6. From the installation directory, modify the file run.bat to point to the location of the SDK runtime .jar file, `StarTeam130.jar`.
For more information, see *Running the Synchronizer* on page 32.

7. Create an ODBC source that points to the Quality Center database.
Bring up the Quality Center Web application in order to download the OTA DLL.

8. Test that the Synchronizer moves change requests from StarTeam into Quality Center, as follows:
   a) Create a change request in StarTeam.
   b) Set the synchronization field in Quality Center to **Yes**.
   c) Run the Synchronizer.
   d) Refresh Quality Center and verify that it created a defect.

9. Test that the Synchronizer moves defects from Quality Center into StarTeam, as follows:
   a) Create a new defect in Quality Center.
   b) Set the synchronization field in Quality Center to **Yes**.
   c) Set the Project field to the appropriate project name.
   Ensure that this field and the parameter **mtd_project** share the same value.
   d) Execute the Synchronizer.
   e) Refresh StarTeam and verify that it created a change request.

10. Run the Synchronizer batch file, `run.bat`, at regular intervals to make the two databases as similar as required by your needs.

When deciding how often to run the batch file, consider the importance of the updates to users versus the possible effects on the performance of the applications, which may slow down during the synchronization process. Use the file `run-again.bat` to set a regular interval, but ensure that the interval is long enough for the Synchronizer to finish the synchronization before it restarts.

Make certain that team members understand the ramifications of the synchronizing decisions. For example, discourage StarTeam users from changing the fields that Quality Center owns because their input will be lost during the synchronization process. Similarly, discourage Quality Center users from changing the fields that StarTeam owns.

As another example, you might tell testers not to change the R&D Comments field and might tell developers not to change the Summary field. Ultimately, you might find it necessary to write procedures for handling defects and change requests in each application to limit mistakes.

Related Topics

Using Quality Center Synchronizer on page 7

Custom Fields

Both Quality Center and StarTeam enable you to create custom fields and make them required. Some custom fields must appear in both applications. For example, you must create a custom field for synchronization purposes in each application. For more information, see *Synchronizing the Applications* on page 8.

It is recommended that custom fields be the same type as the fields to which they are mapped.

Related Topics

Using Quality Center Synchronizer on page 7

Understanding the Synchronizer Properties File

This chapter explains `BugSync.ini`, the properties file for the StarTeam Synchronizer for Quality Center. The file installs as part of the Synchronizer.

Related Topics

Directives

Your properties file contains a series of directives that perform one of the following tasks:

- Identify StarTeam values. These directives begin with `vts`, which represents the StarTeam Server.
- Identify Quality Center values. These directives begin with `mtd`, which represents Quality Center.
- Map Quality Center fields to StarTeam fields or vice versa. These directives begin with `map`.
- (Optional) Map field values from Quality Center to StarTeam or vice versa. These directives begin with `valuemap`. Valuemap fields require that a map value exists for the field itself, as the following example shows:

```
map.BG_Severity>Severity

Valuemap.Bg_Severity.Low=Severity.low
```

🖉 **Note:** Even when a field is mapped from one application to another, mapping its values is not required.

Comments in the BugSync.ini file offer explanations about how to use the directives, which have one of the following formats:

- `key`*n*`=value`

where `key` is the name of the directive plus the required iteration value `n`, and where value is its setting, as the following example shows:

```
vts_server1=localhost
```

For more information, see *Multiple Synchronizations* on page 12.

- **map.**QualityCenterField[>|=|<]StarTeamField

    where QualityCenterField and StarTeamField are a pair of fields between which data will be transferred by the Synchronizer, and where the operator is greater than, equal to, or less than (>, =, or <), as the following example shows:

    For example:

    ```
    map.BG_REPRODUCIBLE>Usr_TDReproducible
    ```

    For more information, see *Quality Center Fields That Contain HTML* on page 26.

- **valuemap.** QualityCenterField.value=StarTeamField.value

    where the value for the QualityCenterField maps to the StarTeamField.value.

    To include special characters, spaces, or operators, the value must be in delimited form. It must also include the quotation marks and parentheses if you want to include spaces in the string, as the follow example shows:

    ```
    QualityCenterField.("value")=StarTeamField.("value")
    ```

Because the properties file must contain user names and passwords for both the Quality Center database and the StarTeam server, you might want to make it inaccessible to others by not sharing the drive on which it resides. If your operating system is Windows, you can hide the folder that contains the properties file by restricting the folder's permissions to a specific group of Windows users. For information about password encryption, see *Passwords* on page 20.

Related Topics

Understanding the Synchronizer Properties File on page 11

Multiple Synchronizations

The initial properties file shows how to synchronize one StarTeam folder with one Quality Center project. However, you can synchronize additional StarTeam folders with additional Quality Center projects in the same properties file.

As stated earlier, `keyn=value` where `key` is the name of the directive plus the required iteration value `n`, and where `value` is its setting.

To identify the first synchronization, you must use directives that end in `1`, such as `vts_project1`, `vts_folder1`, `mtd_driver1`, and so on. Identify the second synchronization with directives that end in `2`, such as `vts_project2`. Directives for the first StarTeam folder must end in `1`, those for the second must end in `2`, and so on. The Synchronizer looks for these index modifier numbers in sequence and stops when it cannot find the next sequential number.

Note: Index modifiers on directives are optional. If you execute only a single iteration, the modifiers for each directive, like `vts_server1` instead of `vts_server`, are not needed.

Any directives that do not end in a number are used for all of the synchronizations that the properties file specifies. Field mappings such as `map.BG_DEV_COMMENTS<Fix+Workaround` and other directives do not

end in numbers and apply to all the synchronizations that use the same properties file. If you need to map fields differently or use different directives, use multiple properties files instead of multiple synchronizations in the same properties file.

⚠️ **Caution:** It is possible to accidentally synchronize the same StarTeam folder's defects multiple times, as might happen when two synchronizations use the same StarTeam project and view and the recursion directive is set to `yes` for both synchronizations. If the synchronizations use the same properties file, it might contain the following directives:

```
vts_recurse_children1=yes
```

```
vts_recurse_children2=yes
```

If the StarTeam folder specified for one synchronization is a child of the StarTeam folder specified for the other, the folder hierarchy for the first folder is recursed twice, which can cause data to appear multiple times in both databases. A similar problem can occur when folders are shared between the two synchronization locations.

When working with multiple synchronizations, you can also use the iterations directive to specify the number of synchronizations in a properties file. Using this directive enables you to define only the required fields that differ from those in the preceding, lower-numbered synchronization.

The Synchronizer writes a file named `<inifile>.<iteration-number>` after each successful synchronization. This file contains the time of the last successful synchronization. If the Quality Center defect and the StarTeam change request have not been changed since that time, the synchronization process skips them during the next synchronization.

Related Topics

Understanding the Synchronizer Properties File on page 11

StarTeam Directives

Use the directives in the following table to identify the StarTeam server, project, and so on. The following table lists them in the order that they appear in the initial properties file, `BugSync.ini`, and in most cases shows how they are initially used in that file.

✏️ **Note:** `<n>` is optional and if not used, the directive applies to all iterations.

StarTeam Directive	Description	Example
`vts_server<n>`	Name or IP address of the computer running a configuration of the StarTeam Server:	`vts_server1=localhost.`
`vts_port<n>`	Port number used by the TCP/IP (sockets) protocol for this StarTeam server configuration. The initial `BugSync.ini` file uses the default port number of 49201.	`vts_port1=49201`
`vts_project<n>`	Name of the StarTeam project to synchronize.	`vts_project1=Mercury`

StarTeam Directive	Description	Example
	The initial `BugSync.ini` file uses `Mercury`, the name of a StarTeam project used for testing with Quality Center.	
`vts_username<n>`	Name of the user whom the Synchronizer uses to log onto StarTeam. This is the user's logon name, not the full name. For example, `jfoulkes` might be the logon name for a user whose full name is `Jessica Foulkes`. At a minimum, the user must possess the following access privileges: • View the project, view, and folder • View, create, and modify the change requests in that folder • Server-level access privileges: "Administer user accounts" and "Change server security setting" are only required if the vts_add_new_enums directive is used or the EnteredBy field is mapped. The Synchronizer generates errors if the mapping directives require StarTeam access privileges that are not granted to the user defined by `vts_username`. If either the `vts_create_custom_fields` or the `vts_add_new_enums` directive is enabled, then the user must also possess the `Add/Modify database schema` access privilege. If `EnteredBy` is mapped and owned by Quality Center, the user must also possess the `Change user/operation time` access privilege. If any of these access privileges are not enabled, an error is generated prior to synchronization, and synchronization is not performed. The initial `BugSync.ini file` uses the generic user name `Administrator`.	`vts_username1=Administrator`
`vts_password<n>`	Password for the user whom `vts_username` defines. The initial `BugSync.ini` file uses	`vts_password1=Administrator`

StarTeam Directive	Description	Example
	`Administrator` because that is the password for the logon name. The initial `BugSync.ini file` uses the password `Administrator` in the sample project.	
`vts_view<n>`	Name of the StarTeam view that stores or will store the synchronized change requests. Ensure that this view is not a read-only view. Leave `vts_view` blank to use the project's root (or default) view. The initial BugSync.ini file leaves this key blank.	`vts_view1=`
`vts_folder<n>`	Path to the StarTeam folder within the StarTeam view, in which the synchronization starts. To use the view's root folder, leave `vts_folder` blank. To use a child of the root folder, specify the path to that folder minus the root folder. Never include the root folder in the `vts_folder`. Use a forward slash (/) as a separator for any other folder names. For example, if the root folder of the view is `StarDraw` and the path to the folder you want to synchronize is `StarDraw/SourceCode/Client`, specify only `SourceCode/Client`. The Synchronizer can keep track of change requests that are moved between StarTeam folders only when the `vts_recurse_children` directive is set to `Yes` and the two folders reside within the hierarchy of the folder specified by the directive `vts_folder`. If these conditions are not met, then new change requests are created instead of the original change requests being moved. The initial BugSync.ini file leaves this key blank to use the view's root folder.	`vts_folder1=`
`vts_recurse_children<n>`	Recurses from `vts_folder` through all of its children. Set to `Yes` or `No`. The initial `BugSync.ini` file uses `Yes`.	`vts_recurse_children1=yes`

StarTeam Directive	Description	Example
vts_send_to_mtd<n>	Name of the custom field in StarTeam that functions as the synchronization flag. This field must be an enumerated field with the case-sensitive values Yes and No. The initial BugSync.ini file uses the field Usr_Sync.	vts_send_to_mtd=Usr_Sync
vts_query	Create a query to filter additional criteria about bugs in StarTeam that should be synchronized.	vts_query=New
vts_use_starteam_user_fullname<n>	Causes the StarTeam full name to be used instead of the logon name. The default is No (and therefore the logon name is used when a StarTeam user field is mapped to a TD string or user field).	

Related Topics

Quality Center Directives

This section describes the directives that identify the Quality Center database, project, and so on. These directives comprise the second set in the properties file, BugSync.ini.

You must specify the following sets of credentials for Quality Center:

- Quality Center database credentials:

    - mtd_username
    - mtd_password

- OTA/Quality Center credentials:

    - mtd_url
    - mtd_app_domain
    - mtd_app_project
    - mtd_app_username
    - mtd_app_password

Additionally, you must specify the OTA/Quality Center credentials to perform updates to Quality Center.

🖉 **Note:** If you are using an Oracle database for your Quality Center projects, your projects might contain Character Large Object (CLOB) fields. To support these fields, modify the directives mtd_driver and mtd_dataurl so that they point to the corresponding Oracle jdbc drivers.

The following table lists the directives in the order that they appear in `BugSync.ini` and shows how that file uses many of them.

Quality Center Directive	Description
`mtd_app_domain`*n*	Quality Center domain. If not specified, `DEFAULT` is used.
`mtd_app_password`*n*	Password for the user whom `mtd_app_username` identifies. This directive is required if a password exists.
`mtd_app_project`*n*	Name of the Quality Center project. This directive is required.
`mtd_app_username`*n*	User name used to log on to the Quality Center project. This directive is required.
`mtd_criteria`*n*	Used with the directive `mtd_project` to build the query that identifies which bugs in Quality Center are synchronized. They can be used independently or together The directive `mtd_project` indicates the value of the `BG_PROJECT` field in Quality Center for the defects to be synchronized. Commenting out this field synchronizes all defects in the Quality Center Project container with StarTeam. The value of `mtd_criteria` is an SQL condition that can restrict the defects referenced for synchronization in Quality Center.
	Users must specify a valid SQL condition. It is recommended that users use the Site Administrator to test the syntax by typing a test `SELECT` clause. For example, the following `SELECT` statement constrains the Synchronizer to defects that are assigned to the user `james_qc`:
	`SELECT * FROM BUG WHERE BG_RESPONSIBLE='james_qc'`
	In this example, the directive `mtd_criteria` is `mtd_criteria=BG_RESPONSIBLE='james_qc'`.
	Unlike `mtd_project`, the Synchronizer ignores the value of this condition when creating new defects in Quality Center.
`mtd_dataurl`*n*	The ODBC DSN or Oracle JDBC URL for the Quality Center database to be synchronized.
	For ODBC, the initial `BugSync.ini` file uses: `mtd_dataurl1=jdbc:odbc:QualityCenter_Demo`
	This declaration identifies the demo database installed with Quality Center. If you create your own Quality Center database, you may also have to create a DSN for it.
	For Oracle JDBC, use one of the following alternatives, which are also shown in the initial `BugSync.ini` file:
	`mtd_dataurl1=jdbc:oracle:thin:@localhost:1521:ORCL`

Quality Center Directive	Description
	`mtd_dataurl1:jdbc:oracle:oci8:@TDORASERVER`
	If you choose the Oracle JDBC URL, you must also use the corresponding Oracle JDBC driver.
`mtd_drivern`	Name of the ODBC or Oracle JDBC driver used to access the Quality Center database.
	Specify this directive for each synchronization, even when the `iterations` directive is used.
	For ODBC, the initial `BugSync.ini` file uses: `mtd_driver1=sun.jdbc.odbc.JdbcOdbcDriver`
	For Oracle JDBC, the initial BugSync.ini file uses the following format: `mtd_driver1=oracle.jdbc.driver.OracleDriver`
	If you choose the Oracle JDBC driver, you must also use the corresponding Oracle JDBC URL.
`mtd_passwordn`	The password for the user whom mtd_username identifies. The initial `BugSync.ini` file leaves this value blank because the user name `admin` has no password in the demo database `mtd_password1=`.
`mtd_projectn`	Case-sensitive value of the field `BG_PROJECT` in Quality Center for the defects to be synchronized, as follows:
	`mtd_project1=Project 1`
	Use the following format to synchronize all Quality Center projects to indicate all values:
	`mtd_projectn=<All Projects>`
	If you comment out the directive, all projects in the Quality Center database are synchronized.
`mtd_send_to_vtsn`	Name of the custom field in Quality Center that functions as the synchronization flag. This field must be a lookup list and must contain the values `Yes` and `No`, which are not case sensitive. The initial `BugSync.ini` file uses the `BG_USER_02` field, as the following example shows:
	`mtd_send_to_vts=BG_USER_02`
`mtd_table_qualifiern`	A user with restricted access to the database can function as the database logon for the Synchronizer. In this case, the table references in the SQL statements must have the Quality Center schema user followed by the owner name of the database schema. The following directive sets the value that is used:
	`mtd_table_qualifier=schema_owner_name`
	For Microsoft SQL Server, the owner is always `td`. For Oracle, the owner is `<domainName>_<ProjectName>_db`.

Quality Center Directive	Description
	The SQL statements that the Synchronizer uses must have the following format: `SELECT * from <mtd_table_qualifier>.BUG` If you do not use this directive, the Synchronizer requires that all tables be owned by the database owner (dbo) instead of Quality Center. As a result, you must run the Microsoft SQL script `set-owner-to-dbo.sql` inside the SQL Server Query Analyzer.
`mtd_url`*n*	Address of the Quality Center server that you enter in a browser, as the following example shows: `http://localhose:8090/qcbin` This required directive allows the use of OTA APIs to write to the Quality Center database.

`mtd_username`*n*	Name of the user that the Synchronizer uses to access the Quality Center database. Because that user name `admin` can access the Quality Center demo database, the initial `BugSync.ini` file uses admin as follows:
	
	`mtd_username1=admin`
	
	With a Microsoft SQL Server database, you might need to set `mtd_username1` to `td`, as shown:
	
	`mtd_username1=td`
	
	`td` is the default user name that Quality Center assigns to any newly created database.
	
	Set `mtd_password` to `tdtdtd`, which is the password used for this database from Quality Center.
	
	With an Oracle database, the user name varies by version. Regardless, the default password is always `tdtdtd`.

Related Topics

Email Messages

Specify the following parameters, which are shown with sample values, to configure the Synchronizer to send an email message whenever a failure occurs during synchronization:

- `email_smtp_host_name=smtp.mybusiness.com`
- `email_smtp_user=myusername`
- `email_smtp_password=mypassword`
- `email_from_address=myusername@mybusiness.com`
- `email_to_address_list=myboss@mybusiness.com`

If you expect the Synchronizer to send an email message outside the SMTP domain, the mail server must support relaying.

Related Topics

Passwords

Passwords are initially entered into the configuration file, `BugSync.ini`, or into a separate file whose name is the name of the configuration file with `.passwords.txt` appended; for example: `bugsync.passwords.txt`.

The passwords are in plain text using the following four directives:

- `vts_password`
- `mtd_password`
- `mtd_app_password`
- `email_smtp_password`

When you run the Synchronizer, the passwords are read from the `.ini` or the `.passwords.txt` file, encrypted, and written to a file whose name is the name of the configuration file with `.passwords.bin` appended; for example: `bugsync.passwords.bin`. After running the Synchronizer, you can remove these passwords from the `.ini` file and comment out the directive. If the passwords are read from a `.passwords.txt` file, the `.passwords.txt` file is deleted after the `.passwords.bin` file is created.

For example, you might begin with the following in the `BugSync.ini` file:

```
vts_password=Administrator
```

The values for ALL passwords written into the `.ini` file (for example, `BugSync.ini`) is stored encrypted in `BugSync.ini.passwords.bin`. If you are changing passwords, you should delete the line in this `.bin` file corresponding to the directive that you changed. Deleting the entire file will require that the passwords be entered in clear text in the `.ini` file. After the first run, the `.bin` file will once again be generated and then the passwords can be removed from the `.ini` file.

After the initial run of the Synchronizer, passwords.bin is created. You can then either remove the second line altogether or comment out the directive `vts_password` and remove or change the value; for example:

```
# vts_password=Password obfuscated
```

OR

```
# vts_password=
```

> **Note:** The Synchronizer uses base64 encoding as well as non-trivial encryption based on elements of the rc4 algorithm. This encryption can be broken because it involves non-secure keys, but such breaking requires decompiling the Synchronizer.

Understanding the Synchronizer Properties File on page 11

Attachments

Use the directives `copy_attachments_from_td_to_st` and `copy_attachments_from_st_to_td` directives to copy attachments.

The following table shows the directives used to copy attachments.

Quality Center Directives	Description
`copy_attachments_from_td_to_st`*n*	Indicates whether attachments found in Quality Center but not found in StarTeam will be copied to StarTeam. The default value is `No`.
`copy_attachments_from_st_to_td`*n*	Indicates whether attachments found in StarTeam but not found in Quality Center will be copied to Quality Center. The default value is `No`.

⚠ **Caution:** If you create a defect in Quality Center with an attachment and then synchronize Quality Center and StarTeam, the new defect and its attachment appear in StarTeam. If you then revisit the defect in Quality Center and replace the original attachment, one of the following events occurs during the next synchronization:

- If the new attachment has the same name as the original attachment, the Synchronizer does not copy the new attachment to StarTeam. Instead, it identifies the attachment as one previously copied to StarTeam and does not recognize that it has changed.
- If the new attachment has a different name, the change request in StarTeam contains both attachments. The Synchronizer does not delete attachments.

Related Topics

Understanding the Synchronizer Properties File on page 11

Custom Fields

The directive `vts_create_custom_fields`*n* determines whether the Synchronizer tries to create a custom field in StarTeam when the mapping for a Quality Center field points to a non-existent StarTeam custom field. Its value can be `Yes` or `No`. The default value is `No`.

When set to `Yes`, the automatically created StarTeam custom field has the name specified in the mapping directive, and its data type is derived from the specified Quality Center field.

This directive does not result in the addition of new enumerating values being added to a StarTeam enumeration. Instead, use `vts_add_new_enums` to accomplish that task. For more information, see *Enumerated Fields* on page 23.

Related Topics

Quality Center Custom Fields

The types for Quality Center fields are number, string, date, user list, or lookup list.

- Number - Maps to StarTeam integer types.
- String - Maps to StarTeam text type fields.
- Date - Maps to StarTeam date and time type fields. StarTeam time-related information is lost in Quality Center. Quality Center date-related information receives a time of midnight (`00:00:00`) in StarTeam.
- User List - Maps to StarTeam user ID type fields.
- Maps to StarTeam enumerated type fields, provided the enumerated values are identical. They do not have to be in the same order.

  If Quality Center owns a lookup list field, you can map it to either a StarTeam custom enumerated field or custom text field. To enable a one-way synchronization, map a Quality Center list field to a StarTeam text field. To enable a two-way synchronization, map a Quality Center lookup list field to a StarTeam enumerated field.

For a list, check **Verify Value** to restrict the field to the values in the list. For more information about customizing fields, see your Quality Center documentation and Help.

Related Topics

StarTeam Custom Fields

The following list describes the field types for StarTeam.

- Dates/Times - StarTeam date and time type fields can be matched to Quality Center date type fields. These formats are set by using the **Regional and Language Options** dialog box, which is accessible through the **Control Panel**. If StarTeam does not own the field, time information that is missing from Quality Center is lost.
- Text - StarTeam text type fields can be mapped to Quality Center string type fields. Length limitations in Quality Center string fields and list items are database dependent. For more information, see the *Quality Center Administrator's Guide*. StarTeam text fields must be similarly limited or Quality Center must own the field.
- Integers and Real Numbers - Quality Center owns no real type fields. The StarTeam change request component also owns no real number fields unless someone creates a custom field of this type. If appropriate, a user can map StarTeam fields that contain real numbers to a Quality Center text field.

  StarTeam integer type fields can be mapped to a Quality Center number type.

- Enumerations - StarTeam enumerated fields can be mapped to a Quality Center lookup list field.

  Strings represent enumerated values in Quality Center. The length limitations of these strings depend upon the database used for Quality Center. Quality Center enumeration values are limited to 40 characters. To map an enumerated StarTeam field with a name that is longer than the Quality Center limit, edit the field

so that it is less than or equal to that length, and then use the same values for the mapped Quality Center lookup list field.

- User ID - If a user list matches StarTeam logon names for users, StarTeam user ID fields can be mapped to a Quality Center user list field.

For more information about customizing fields, see the *Borland StarTeam Cross-Platform Client Help*.

Related Topics

Custom Fields on page 21

Blank Fields

The Synchronizer can transfer data from a field that is blank to a field with a value. StarTeam supports blank fields, and Quality Center list fields can accept a blank value, unless the **Verify Value** option is selected.

The following mapping in a properties file indicates that the Quality Center **Detected On Date** field, **BG_DETECTION_DATE**, is mapped to a StarTeam custom field named **Usr_TDDetectionDate**:

```
map.BG_DETECTION_DATE>Usr_TDDetectionDate
```

If the **Detected On Date** is initially set to **1/1/00** in Quality Center, then the process of running the Synchronizer results in this value being added to StarTeam. If you later delete this value from Quality Center by making the **Detected On Date** field blank, then the process of running the Synchronizer results in the StarTeam field becoming blank as well.

Related Topics

Understanding the Synchronizer Properties File on page 11

Enumerated Fields

The synchronization of enumerations between StarTeam and Quality Center is a two-way process. The following directives, both of which default to `No`, control this functionality.

- `vts_add_new_enums`
- `mtd_add_new_enums`

When `vts_add_new_enums` is set to `Yes`, enumeration values are added to StarTeam if they are defined in the Quality Center list but not defined in StarTeam.

When `mtd_add_new_enums` is set to `Yes`, enumeration values are added to Quality Center if they are defined in the StarTeam list but not defined in Quality Center.

These directives apply equally to StarTeam fields that are of type view label, such as the `LastBuildTested` and `AddressedIn` fields, and of type view name, such as the `AddressedInView` field. For example, if `vts_add_new_enums` is set to `Yes`, `BG_USER_06` is mapped to `LastBuildTested`, and `BG_USER_06` has a value of `Not Yet Tested`, the Synchronizer attempts to create a build label named `Not Yet Tested`.

These directives do not work for StarTeam fields or Quality Center fields that are user IDs because the Synchronizer does not add users to either StarTeam or Quality Center. Instead, the Synchronizer synchronizes between user ID fields, although it does not modify the lists of users in either StarTeam or Quality Center.

To apply either directive to specific fields in Quality Center or StarTeam, specify a list of fields that are separated by commas. A value of `Yes` is assumed. For example, the following directive adds new Quality Center enumerations only to the fields `BG_USER_06` and `BG_USER_07`:

```
mtd_add_new_enums= BG_USER_06, BG_USER07
```

This approach is useful when a Quality Center enumeration maps to a StarTeam build label field like `LastBuildTested`, adding new StarTeam build labels to the enumerations in Quality Center.

When `mtd_add_new_enums` is enabled, the Synchronizer adds values from StarTeam to Quality Center. When the StarTeam field is either `LastBuildTested` or `AddressedIn`, the number of values involved can be large. Unlike StarTeam, which stores enumeration values in a list, Quality Center stores enumeration values in a tree.

Quality Center possesses a limit of 677 children for any node in the tree. If the number of elements in the Quality Center enumeration is small, the Synchronizer adds top-level nodes to the enumeration. However, if adding the new elements causes the number of elements in the enumeration to be greater than one-half of 677, or 338, the Synchronizer finds the parent node whose description is most common to the description of the value being added. If a node with any commonality exists, the new node is added as a child node. Otherwise it is added as a top-level node. During a single synchronization, nodes are added in sorted order so that a tree can be generated based on the above algorithm.

The following structure represents a typical tree structure for build labels:

- Build 1

  - Build 1.1
  - Build 1.2

    - Build 1.2.1

- Build 2

...

Related Topics

Mapping Enumerated Fields to Text Fields

You can map enumerated fields to text fields if you take precautions. For example, the application in which the enumerated field resides must own the mapping because a text field can contain data that does not match the enumerated value available in the field receiving the data. Text might also be misspelled, possess too many characters, or not appear in the enumerated values.

Related Topics

Empty Enumerations

If the Quality Center value is empty and mapped to a StarTeam enumeration, the default value of the enumeration in StarTeam is used. Even if Quality Center owns the mapping, the StarTeam change request is updated with the default value when the value in Quality Center is empty. When the next synchronization is performed, those default values from StarTeam will be copied into the fields in Quality Center.

If a Quality Center field allows blank values and maps to a StarTeam enumeration, a warning instructs the user to make the Quality Center field `Required` in Quality Center, thus preventing a blank value.

Related Topics

Field Properties for Quality Center Enumerated Fields

Related Topics

Verify Value

For any Quality Center defect, if an enumerated field has **Verify Value** set to `True`, the field cannot have a value that is not in the list of enumerations for that field. If **Verify Value** is set to `False`, the field can contain a value different that those in the list.

Note: The new value is not added to the list. Instead, it is used as the value of the field in the given defect.

If the following statements are true for a defect that is being synchronized, the update will not occur and an error will be logged:

• The **Verify Value** property is set for a Quality Center field in a Quality Center defect.
• The Synchronizer needs to update that field in that defect.
• The corresponding StarTeam field contains a value that is **not** in the list of possible values for the Quality Center field.

Allow Multiple Values

For any Quality Center defect, if an enumerated field has **Allow Multiple Values** set to `True`, the user can select more than one value from the list of enumerations for that field.

A multi-valued enumeration in Quality Center must be mapped to a StarTeam text field. When filled by the Synchronizer, the StarTeam field mapped to a multi-valued enumeration contains all the selected values, using a semicolon as a delimiter.

StarTeam Change Requests and Correct Iterations

When multiple iterations are defined, the Synchronizer detects if a StarTeam change request must belong to a different iteration based on its associated Quality Center properties. In such cases, the change request is moved in StarTeam from its current location to the root folder of the correct iteration.

Because all of the change requests must be collected before any synchronization can be performed, this functionality can have significant memory overhead. The move functionality is turned on by default. The following command turns off the move functionality:

```
VTS_ENABLE_MOVE_CR_CAPABILITY=NO
```

If the move functionality is required, the Synchronizer and server memory can be held to a minimum, at the cost of slower performance and server overhead, by keeping a single StarTeam view open at a time. During change request collection, each view is sequentially opened, read, and then closed. During synchronization, each view is opened again.

The option to keep a single view open at a time is turned on by default. The following command turns off this option:

```
VTS_KEEP_ONLY_ONE_VIEW_OPEN=NO
```

Setting this option to `No` allows the Synchronizer to move change requests in StarTeam faster, but at the expense of both client and server memory.

If you use `VTS_ENABLE_MOVE_CR_CAPABILITY` to turn off the move functionality, then the number of views opened at one time has a smaller effect on memory. Additionally, the `VTS_VIEW_OPEN` option is ignored, regardless of its setting. Both options are ignored if only a single view is referenced, regardless of the number of Iterations.

Related Topics

Quality Center Fields That Contain HTML

The `vts_remove_html`n directive provides for the mapping of Quality Center fields that support HTML to StarTeam fields in plain text. The comparison and conversion processes between HTML and StarTeam recognize `<br>` as a carriage return/line feed. The same processes recognize eight spaces in Quality Center as a tab in StarTeam.

The `vts_remove_html`n directive can be set to `Yes` or `No`. The default value is `Yes`. The mapping between the two fields can be >, =, or <.

Quality Center Field Defect ID Value Format

The value of the `vts_bug_id_format` directive formats the value of the Quality Center field `Defect ID`, whose internal identifier is `BG_BUG_ID`. Without the directive value, the StarTeam field that maps to the field `Defect ID` is formatted using the format `TD BUG #: %d`, where `%d` is the number of the Quality Center defect.

Use the directive `vts_bug_id_format` to change this format. For example, the following directive pads the left six digits of the resulting number with zeros:

`vts_bug_id=QualityCenter Bug: %1$06d`

This directive produces the following result:

`QualityCenter Bug: 000001`

Because the directive `vts_bug_id_format` is not used when parsing the value, the format must not contain any digits other than those from the Quality Center defect number. For example, do not use the format: `1st Set of Defects: %d`.

Mapping Directives

As the Synchronizer transfers defect-related information from one database to the other, it copies data from a field in one application to a field in the other application. The properties file contains mapping directives that will contain the same data after the synchronization. Each mapping directive specifies a pair of fields, one from Quality Center and one from StarTeam; for example: `map.QualityCenterField [>|=|<]StarTeamField`.

A valid mapping directive must satisfy the following criteria:

- Begin with `map` or `valuemap`.
- The Quality Center field always appears first in the mapping.
- Identify the Quality Center field by its internal identifier.

  For a Quality Center field, the internal identifier is the name in the Quality Center database table.

- Identify the StarTeam field by its internal identifier.

  For a StarTeam field, the internal identifier is the name by which StarTeam Server recognizes the field. For more information, see *Appendix A - StarTeam Change Request Fields* on page 39.

- Separate the fields with one of the following operators to indicate the direction of the data flow.

- > represents a mapping from Quality Center to StarTeam in which Quality Center owns the pair of mapped fields. Use with map directives only.
- < represents a mapping from StarTeam to Quality Center in which StarTeam owns the pair of mapped fields. Use with map directives only.
- = indicates that the Synchronizer uses the dates in the histories for a set of paired fields to determine the value to place in both fields. Quality Center History must be enabled on the field for the operator to work.

`BugSync.ini`, the initial properties file included with the installation, is shown in *Appendix C - Initial Synchronizer Properties File* on page 60.

It is recommended that you perfom all of the following steps:

1. Create a copy of the file `BugSync.ini`.
2. Rename the file.
3. Edit the file to meet your needs.
4. Remove the comment symbols (#) where appropriate to activate mapping directives.

The Synchronizer uses either ownership or field history to determine what data is copied to the respective databases.

Related Topics

Understanding the Synchronizer Properties File on page 11

Ownership (How the Operators Work)

When either Quality Center or StarTeam owns the pair of mapped fields, those fields are always synchronized to match each other, based on the value of the field in the application that owns the pair. Ownership is shown by using a greater than (>) or less than (<) operator between the fields' internal identifiers in the mapping directive.

The following example shows the Quality Center field `BG_DESCRIPTION` mapped to the StarTeam field `Description`.

```
map.BG_DESCRIPTION>Description
```

The operator > indicates that Quality Center owns the pair of mapped fields As a result, the value in the Quality Center field can be copied to the StarTeam field during synchronization, but that the value in StarTeam cannot be copied to Quality Center.

During synchronization, any changes made to the Quality Center `Description` field are transferred to the StarTeam `Description` field. All changes made to the StarTeam `Description` field since the last synchronization are overwritten. Between synchronizations, the two fields can contain different data. After a synchronization, however, they both contain the data found in the Quality Center `Description` field.

The following example shows the Quality Center field `BG_DEV_COMMENTS` mapped to the StarTeam field `Fix`:

```
map.BG_DEV_COMMENTS<Fix
```

The operator < indicates that StarTeam owns the pair of mapped fields. As a result, the value in the StarTeam field can be copied to the Quality Center field during synchronization, but the value in Quality Center cannot be copied to StarTeam.

Mapped fields can be updated based on the time at which their values were last updated. The Synchronizer uses each field's history to determine the direction that the data will flow. In the mapping directive, the fields'

histories are used whenever the operator = separates the internal identifiers for each field. As a result, the value in the Quality Center field can be copied to the StarTeam field during synchronization, or vice versa, depending on which value is more recent.

The following example shows the Quality Center field `BG_RESPONSIBLE` mapped to the StarTeam custom field `Usr_TDResponsible`.

```
map.BG_RESPONSIBLE=Usr_TDResponsible
```

To use historical data, select the History option for the Quality Center field. You do not need to change anything in StarTeam because StarTeam automatically stores historical data about each field.

If you plan to use history to control data exchange, all computers running Quality Center and StarTeam must have synchronized clocks.

If the operator = separates the names of two mapped fields in the properties file, their histories determine what value appears in that field for each application.

The Quality Center `Description` and `R & D Comments` fields must use ownership when they are mapped because they do not support history.

Requirements and Limitations of Mapping

The following mappings are required:

- Map the Quality Center field `Defect ID` to the StarTeam field `External Reference` with Quality Center as the owner. This following example shows the text of this mapping as it appears in the properties file:

  ```
  map.G_BUG_ID>ExternalReference
  ```

- StarTeam Quality Center Synchronizer requires the following directives:

  - `mtd_url`
  - `mtd_app_domain`
  - `mtd_app_username`
  - `mtd_app_password`

  If you do not specify the directive `mtd_app_domain`, `DEFAULT` is used.

The following mappings are recommended:

- Map the Quality Center field `R & D Comments` to either or both of the StarTeam fields `Fix` or `Workaround` with StarTeam as the owner. The following examples show the text of these mappings as they appears in the properties file:

  ```
  map.BG_DEV_COMMENTS<Fix
  ```

  ```
  map.BG_DEV_COMMENTS<Workaround
  ```

  ```
  map.BG_DEV_COMMENTS<Fix+Workaround
  ```

  Do not use the operator = in this mapping because Quality Center cannot store history for its `R & D Comments` field.

- If you plan to extract defects from Quality Center to StarTeam, map the Quality Center field `Summary` to the StarTeam field `Synopsis`, with Quality Center as the owner. The following example shows the text of this mapping as it appears in the properties file:

```
map.BG_SUMMARY>Synopsis
```

- If you plan to extract defects from Quality Center to StarTeam, map the Quality Center field `Description` to the StarTeam field `Description` with Quality Center as the owner. The following example shows the text of this mapping as it appears in the properties file:

```
map.BG_DESCRIPTION>Description
```

Do not use the operator = in this mapping because Quality Center cannot store history for its `Description` field.

The following mappings are restricted:

- All StarTeam user fields can map to Quality Center fields. However, the mapping for the StarTeam field `EnteredBy` can be < or >. Setting the mapping to = generates an error. If the mapping is <, the field in StarTeam can only be set when the CR is created, if the mapping is > then the field in TD will be updated anytime the value in TD isn't the same as the value in StarTeam.

    If a change request is created and Quality Center owns the mapping, then the value from Quality Center sets the StarTeam field `EnteredBy`. The Synchronizer accomplishes this task by changing user IDs. The user whom the directive `vts_username` defines must possess the **Change user/operation time** access privilege that the Server Administration Tool defines. If StarTeam owns the mapping, then the field `EnteredBy` is set to the value in `vts_username`.

- The Synchronizer supports mapping to and from StarTeam real values. If the operator = is used, the value must map to a string because Quality Center doesn't support real values. If StarTeam owns the mapping, then a Quality Center number field can be used, in which case the value is truncated when written to Quality Center.

- The following table lists StarTeam fields that cannot be mapped unless StarTeam owns the pair of mapped fields.

StarTeam Fields	Internal Identifiers
Addressed By	AddressedBy
Addressed In Build	AddressedIn
Branch On Change	BranchOnChange
Branch State	BranchState
Comment ID	CommentID
CR Number	ChangeNumber
Created By	CreatedUserID
Created Time	CreatedTime
Dot Notation	DotNotation
Dot Notation ID	DotNotationID
End Modified Time	EndModifiedTime
Entered On	EnteredOn
Last Build Tested	LastBuildTested
Modified By	ModifiedUserID
Modified Time	ModifiedTime

StarTeam Fields	Internal Identifiers
Object ID	ID
Parent Branch Revision	ParentRevision
Parent ID	ParentObjectID
Parent Revision	PathRevision
Read Only	ReadOnly
Resolved On	ResolvedOn
Root Object ID	RootObjectID
Share State	ShareState
Verified On	VerifiedOn
Version	RevisionNumber

- The following table lists Quality Center fields that cannot be mapped unless Quality Center owns the pair of mapped fields.

Quality Center Fields	Internal Identifiers
Defect ID	BG_BUG_ID
Reproducible	BG_REPRODUCIBLE
Subject	BG_SUBJECT
Test Set	BG_CYCLE_REFERENCE

- Comment or leave out mappings that you do not want.

Running the Synchronizer

Related Topics

Synchronizing Between Servers in Different Time Zones on page 34

Batch Files

If you are using the StarTeam Windows Server, the StarTeam Quality Center Synchronizer installation includes files `run.bat` and `run-again.bat`.

Use the file `run.bat` to run the Synchronizer at any time. The file `run-again.bat` enables you to run the Synchronizer at regular time intervals by setting the number of seconds to wait. The wait setting is found at the end of the file.

You might need to modify the original `.bat` files to point to the locations that your organization uses for the following installations and libraries:

- The StarTeam installation. The initial batch files include the following line:

  `SET STARTEAM_Path=C\Program Files\Borland`

- The StarTeam SDK installation. The initial batch files include the following line:

  `SET STARTEAM_SDK_PATH=%STARTEAM_PATH%\StarTeam`

  `SDK xx\`

- The StarTeam SDK library. The initial batch files include the following line:

  `SET SDK_JAR=%STARTEAM_SDK_PATH%\Lib\StarTeamxx.jar`

- The Oracle Client installation. The initial batch files include the following line:

  `SET ORACLE_PATH=C:\oracle\ora92`

- The Oracle JDBC library. If you are using an Oracle database for Quality Center projects, your projects might contain CLOB fields. Correctly specify this path to support these fields, and modify the directives mtd_drivern and mtd_dataurln to point to the corresponding Oracle JDBC drivers.

  The initial batch files include the following line:

  `SET ORACLE_CLASSPATH=%ORACLE_PATH%\jdbc\lib\classes12.zip`

- The Java lib path. The initial batch files include the following line:

  `SET _JAVA_LIB_PATH=%STARTEAM_SDK_PATH%\Lib;%Oracle_Path%\bin`

- The StarTeam SDK Installed Java Runtime Environment. The initial batch files include the following line:

  `SET JAVA="%STARTEAM_PATH%\Java\Sun1.6.0_02\bin\java.exe`

To view the contents of the batch files, see *Appendix D - Initial Synchronizer Batch Files* on page 69.

Log Files

The Synchronizer reports errors and other information in `mtdsync.log` and `mtdsync.err`. For example, `mtdsync.log` contains information about when you started `run.bat`, and `mtdsync.err` lets you know when data has been truncated.

Synchronizer Execution Line Options

The table below lists the options that can appear at the end of the Synchronizer execution line.

Option	Action	Example
`log <directory>`	Generate the `mtdsync.log` and `mtdsync.err` files in a directory other than the current directory.	`%JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"com.starbase.mtdsync.App BugSync.ini log "c:\inetpub\wwwroot\tdlogs"`
`debug`	Generate a more detailed `mtdsync.log` file.	`%JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"com.starbase.mtdsync.App BugSync.ini debug`
`Net Monitor`	Displays the output by Net Monitor.	`%JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"com.starbase.mtdsync.App BugSync.ini netmon`
`nobuffer`	Turn off buffering and enable real-time logging. This change slows down the Synchronizer.	`%JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"com.starbase.mtdsync.App BugSync.ini nobuffer`
`readonly`	Perform a test run of the Synchronizer without updating any data. This option can be used with or without the `debug` option.	`%JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"com.starbase.mtdsync.App BugSync.ini readonly`
`repeat n`	Perform a test run of the Synchronizer to loop for the number of seconds defined until **Enter** is pressed in the **Command Prompt** window. This option can be used with or without the `postSyncBat<filepath>` option. ⚠️ **Caution:** When handling a large amount of data, using this option without	`REM %JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%" com.starbase.mtdsync.App BugSync.ini repeat 5 postSyncBat "C:\Program Files\Borland\StarTeam QualityCenter Synchronizer\postSync.bat"`

Option	Action	Example
	`postSyncBat<filepath>` will result in a massive log file.	
`postSyncBat<filepath>`	Used with the repeat option, this option calls a batch file to execute after each synchronization.	

Synchronizing Between Servers in Different Time Zones

If the Synchronizer machine and the Quality Center server run in different time zones, use the directive `mtd_timezone` to specify a time zone ID code for the location of the Quality Center server; for example: `mtd_timezone1=US/Hawaii`.

Specify an invalid code to view a list of available codes. Common codes include `US/Pacific`, `US/Arizona`, `US/Central`, and `US/Eastern`.

Related Topics

Issues for Changing the Separation Between the Servers

When the Quality Center server and the machine running the Synchronizer reside in the same time zone, no problems arise. Additionally, no problems arise if your properties file does not contain any == mapping fields that compare times.

Quality Center does not store dates in GMT. If you move either the Quality Center server or the machine running the Synchronizer to a new time zone, you must run the Synchronizer before the move and update the directive `mtd_timezone` in the file `BugSynch.ini.`

If you cannot synchronize before the move, you must wait the number of hours equal to the time difference between the two time zones before you synchronize the databases again, so the timestamp on the latest record change in either database has time to occur in the earlier time zone.

Related Topics

Example

If your properties file contains the following mapping:

`BG_SUMMARY == Synopsis`

and if you synchronize in Los Angeles, California, and your server is in Los Angeles,

At 1:00 PM PST

ST: Syn -> v1 (timestamp is 1:00 PM)

Sync at 1:30

TD: BG_SUMMARY -> v1 (timestamp is 1:30 PM)

Move the Quality Center server to Hawaii (-4 hours from PST)

At 2:00 PM PST

ST: Syn -> v2 (timestamp is 2:00 PM)

Sync at 2:30 PM

The 1:30 PM timestamp on the field `BG_SUMMARY` is viewed as being 5:30 PM PST, which is later than the 2:00 PM `Synopsis` timestamp:

```
BG_SUMMARY --> Syn
```

However, if you wait four hours before modifying Synopsis, the StarTeam timestamp will be greater than the `BG_SUMMARY` timestamp.

Related Topics

Troubleshooting

This chapter contains miscellaneous information about issues that can arise as you use the Synchronizer.

Related Topics

StarTeam Issues

The topics in this section contain information about StarTeam issues that can arise as you use the Synchronizer.

Related Topics

StarTeam Does Not Display Quality Center Defects

If you are using an existing Quality Center database (such as the QualityCenter_Demo database that installs with Quality Center), none of the Quality Center defects will appear in StarTeam until the Quality Center synchronization flag for those defects is set to Yes.

The reverse is true in StarTeam databases if the synchronization field defaults to Yes. All change requests appear in Quality Center unless you set the field value to No. Note that the change requests are not removed, but are no longer updated.

Related Topics

Fields Not Updated

If a memo field intended for a StarTeam change request exceeds the length limitations of the Quality Center database, a warning message appears in the log file, and the change request field is not updated.

Error Messages

Do not use a read-only StarTeam view. If you specify a read-only view for the directive `vts_viewn` (where *n* is an iteration value, such as `1`) in the properties file, the following error message appears in `mtdsync.err` when you attempt to connect to the StarTeam Server:

```
java.net.ConnectException: Connection refused
java.lang.RuntimeException: java.net.ConnectException: Connection refused
```

Quality Center Issues

The topics in this section contain information about StarTeam issues that can arise as you use the Synchronizer.

Truncated Fields

Memo fields are large text or string fields that can contain from 2K to 20K of data. The fields **Description** and **R & D Comment** are examples of Quality Center memo fields, and the fields **Description**, **Fix**, and **Work Around** are examples of StarTeam memo fields.

In general, StarTeam memo fields can contain more characters than Quality Center fields. The number of characters in Quality Center and StarTeam memo fields might differ because their length depends on the database being used. If a field length does not approach the specific database limit, no problem occurs. If both applications use the same database version, however, they might possess the same field-length limitations. For additional information on field length, see the *Quality Center Administrator's Guide*.

When the data from a StarTeam text field exceeds the length limitations for the Quality Center field that receives the data, the files `mtdsync.log` and `mtdsync.err` log a warning, and the value of the field is truncated as it is added to Quality Center. The Synchronizer does not set an error level that the batch file can detect.

If you create a Quality Center defect with a memo field that exceeds 250 characters, a SQL limitation results in an empty defect. No warning message appears in the log file because the field might be filled when the defect is next updated.

The Synchronizer reads the lengths of the Quality Center fields each time it runs. Current versions of Quality Center are flexible enough to operate on tables whose columns have been increased in size through direct access to the database. The Synchronizer recognizes the new field sizes and adjusts accordingly.

If the data from a Quality Center field exceeds the length limitations for the StarTeam field, the value of the field is truncated and a StarTeam database exception is recorded. A warning with an error value is returned to the batch file, and the stack trace of the exception is recorded in the files `mtdsync.err` and `mtdsync.log`.

All warning messages in the `mtdsync.err` file indicate the numbers for both the Quality Center and StarTeam defects currently being processed. In the following example, a StarTeam text field that contains 47 characters is truncated to 40 characters:

```
Error Report:
1. TD Bug:72 ST Bug:46 MTD column BG_USER_05 has been truncated from 47 to 40
characters.
```

Related Topics

Quality Center Issues on page 37

Empty Fields

if you update an existing Quality Center defect with a memo field that exceeds the length limitations for the Quality Center database, the specific field is truncated, and the action is logged in the file `mtdsync.log`.

Related Topics

Quality Center Issues on page 37

Appendix A - StarTeam Change Request Fields

This section explains StarTeam change request fields and suggests mappings between change request fields and Quality Center defect fields. Change requests are equivalent to Quality Center defects. In StarTeam, you must use the **New Change Request** dialog box to add change requests manually.

The text *(Advanced)* follows the names of StarTeam Advanced fields, which are listed for completeness. However, most users rarely use advanced fields. The fields `Locked By`, `My Lock`, and `Non-Exclusive Lockers` have been omitted because change requests cannot be locked.

- **Addressed By**

  Type: User ID.

  Value: Either a selection from a list of users or `<None>`.

  Internal Identifier: `AddressedBy`.

  Description: Indicates the name of the user who resolved the change request. Change requests with the following states are considered resolved:

  - Cannot Reproduce
  - As Designed
  - Fixed
  - Documented
  - Is Duplicate

- If the Quality Center field `User List` uses a list that matches the StarTeam user logon names and if it requests verification, you can map it to the StarTeam field `Addressed By`. However, the length of user names in Quality Center is database dependent, while StarTeam allows up to 63 characters. StarTeam must own this mapping.

- **Addressed In Build**

  Type: Enumerated.

  Value: Either a selection from a list of view labels or `<None>`.

  Internal Identifier: `AddressedIn`.

  Description: The next build label created and applied to a view after a change request is resolved.

  You can map this field to a Quality Center string field or lookup list field that matches the StarTeam labels and that requests verification. StarTeam must own this mapping.

- **Addressed In View**

  Type: Enumerated.

  Value: Either a selection from a list of views or `<None>`.

  Internal Identifier: `AddressedInView`.

  Description: Indicates the view in which the change request was resolved. This field is important for shared and for moved change requests.

When StarTeam owns the mapping (<), you can map this field to a Quality Center string field or a lookup list field. When the mapping is a two-way synchronization (=), map this field to a Quality Center lookup list field that has a matching list of values.

- **Attachment Count**

  Type: Integer.

  Value: Database dependent.

  Internal Identifier: `AttachmentCount`.

  Description: Number of files attached to a change request.

  You can map this field to an integer field in Quality Center.

- **Attachment IDs** (Advanced)

  Type: Text.

  Value: Byte array displayed as a bracketed series of numbers in hex format. For example, `[00 00 00 00 02 00 00 00]` indicates two specific attachments.

  Internal Identifier: `AttachmentIDs`.

  Description: ID numbers assigned to attachments.

  Do not map this field because attached files cannot be moved from one system to another.

- **Attachment Names**

  Type: Text.

  Value: File names separated by spaces.

  Internal Identifier: `AttachmentNames`.

  Descriptions: Lists the names of the files attached to a change request. Spaces separate items in the list.

  Do not map this field because attached files cannot be moved from one system to another.

- **Branch On Change** (Advanced)

  Type: Enumerated.

  Value: `No` or `Yes`.

  Internal Identifier: `BranchOnChange`.

  Description: Indicates whether a change request branches when it changes. If the value is `No`, the change request's behavior is not set to `Branch On Change` for any of the following reasons:

  - The change request is in the root or a reference view and the `Branch On Change` feature is disabled.
  - The change request is in a branching view but has already branched as a result of a change. This event disables the `Branch On Change` feature.
  - The change request is in a branching view, but its behavior currently does not permit it to branch on change. As a result, modifications are checked into the parent view.

    ✎ **Note:** If the value is `No`, the value of the Branch State explains why.

    The value `Yes` indicates that the change request resides in a branching view and has its behavior set to `Branch On Change`, but has yet to be changed.

You can map this field to a Quality Center string field or lookup list field that has the values `No` and `Yes` and that requests verification. StarTeam must own this mapping.

- **Branch State** (Advanced)

    Type: Enumerated.

    Value: `Branched`, `Not Branched`, or `Root`.

    Internal Identifier: `BranchState`.

    Description: Indicates whether a change request has branched in the child view, is still unbranched and, therefore, a part of the parent view, or was created in the view in which it resides.

    The values `Branched` and `Not Branched` apply to change requests in branching views. The value `Root` applies to files created in the view in which the change request currently resides.

    If the view is a reference view, it reflects the state of the change request in the reference view's parent.

    You can map this field to a Quality Center string field or lookup list field that has the values `Branched`, `Not Branched`, and `Root` and that requests verification. StarTeam must own this mapping.

- **Category**

    Type: Text.

    Value: Up to 64 characters.

    Internal Identifier: `Category`.

    Description: Text identifying the subcomponent in which the defect occurs. It is typically used in combination with the field `Component`.

    You can map this field to a custom Quality Center string field. However, the maximum length of Quality Center string fields is database dependent. As a result, Quality Center must own this field to prevent loss of data.

- **Closed On**

    Type: Date/Time.

    Value: Date and time.

    Internal Identifier: `ClosedOn`.

    Description: Date and time when a change request was closed. StarTeam automatically completes this field as the change request's status becomes closed.

    If StarTeam owns the pair of mapped fields, you can map this field to a or Quality Center date field. However, the time will be lost.

- **Comment**

    Type: Memo (very large string field).

    Value: Very large number of text characters.

    Internal Identifier: `Comment`.

    Description: Contains the full text of the revision comment that explains the reason for modifying a change request's properties. The first 2000 characters of the revision comment are stored in the `Short Comment` field. The process of modifying the properties causes StarTeam to create a new revision.

    You can map this field to a Quality Center string field. However, the data might be truncated when it is exported to Quality Center because the maximum length of Quality Center string fields is database dependent.

- **Comment ID** (Advanced)

Type: Integer.

Value: Database dependent.

Internal Identifier: `CommentID`.

Description: ID number that is assigned to a revision comment. If a revision comment is not supplied, this field displays a value of -1.

If StarTeam owns the pair of mapped fields, you can map this field to a for Quality Center string field.

- **Component**

    Type: Text.

    Value: Up to 64 characters.

    Internal Identifier: `Component`.

    Description: Text identifying the component in which the defect occurs. It is often used with the field `Category` to narrow that identification to a subcomponent.

    You can map this field to a custom Quality Center string field. However, the maximum length of Quality Center string fields is database dependent. As a result, Quality Center must own this field to prevent loss of data.

- **Configuration Time**

    Type: Date/Time.

    Value: Date and time.

    Internal Identifier: `ConfigurationTime`.

    Description: Indicates the date and time to which a change request is configured. If you configure a change request to a specific time, this field contains that time. If you configure a change request to a label or promotion state, this field shows either the time at which the label was created or the time at which the label associated with the promotion state was created.

    If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center date field. However, the time will be lost.

- **CR Number**

    Type: Integer.

    Value: Database dependent.

    Internal Identifier: `ChangeNumber`.

    Description: Number assigned to a change request. For example, if the `Object ID` is `0`, the change request number is `1`.

    StarTeam controls this field. If StarTeam owns the pair of mapped fields, you can map it to a Quality Center number field.

- **Created By** (Advanced)

    Type: User ID.

    Value: Either a selection from a list of users or `<None>`.

    Internal Identifier: `CreatedUserID`.

    Description: Name of the user who created the first revision in the view. This value is either the user who initiated the change request or the user who modified the revision that branched.

If the Quality Center field uses a list that matches the user logon names in StarTeam and that requests verification, you can map this field to a Quality Center User List field. However, the length of user names in Quality Center is database dependent, while StarTeam allows up to 63 characters. StarTeam must own this mapping.

- **Created Time** (Advanced)

  Type: Date/Time.

  Value: Date and time.

  Internal Identifier: `CreatedTime`.

  Description: Date and time at which the first revision in the view was created.

  If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center date field. However, the time will be lost.

- **Deleted By**

  Type: User ID.

  Value: Either a selection from a list of users or `<None>`.

  Internal Identifier: `DeletedUserID`.

  Description: Name of the user who deleted a change request. This information is unavailable to users because deleted change requests do not appear in the list.

  This field is not available for mapping.

- **Deleted Time (Advanced)**

  Type: Date/Time.

  Value: Date and time.

  Internal Identifier: `DeletedTime`.

  Description: Date and time when a change request was deleted. This information is unavailable to users because deleted change requests do not appear in the list.

  This field is not available for mapping.

- **Description**

  Type: Memo (very large string field).

  Value: Very large number of text characters.

  Internal Identifier: `Description`.

  Description: Description of the problem, including the steps needed to reproduce it.

  It is recommended that you map the Quality Center field `Description` to the StarTeam field `Description`, with Quality Center as the owner of the pair.

- **Dot Notation**

  Type: Text.

  Value: Branch revision number.

  Internal Identifier: `DotNotation`.

  Description: Branch revision number, like `1.2.1.0.`

  If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center string field.

- **End Modified Time** (Advanced)

Type: Date/Time.

Value: Date and time.

Internal Identifier: `EndModifiedTime`.

Description: Date and time when a revision ceased to be the tip revision. Although the upper pane of a StarTeam client can display this field, its value is always blank because the item remains the tip revision at any given configuration time.

If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center date field. However, the time will be lost.

- **Entered By**

    Type: User ID.

    Value: Either a selection from a list of users or `<None>`.

    Internal Identifier: `EnteredBy`.

    Description: Name of the user who created this change request.

    If the Quality Center field uses a list that matches the user logon names in StarTeam and that requests verification, you can map the StarTeam field `Entered By` to a Quality Center user list field. However, the length of user names in Quality Center is database dependent, while StarTeam allows up to 63 characters. Either application can own this mapping. However, if Quality Center owns the mapping, it affects the StarTeam user shown in the field `Entered By`.

- **Entered On**

    Type: Date/Time.

    Value: Date and time.

    Internal Identifier: `EnteredOn`.

    Description: Date and time when this change request was created.

    If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center date field. However, the time will be lost.

- **External Reference**

    Type: Text.

    Value: Up to 64 characters.

    Internal Identifier: `ExternalReference`.

    Description: Indicates a customer or other outside source who provided the data for this change request.

    This field must map to the Quality Center field `Defect ID` (BG_BUG_ID). Quality Center must own the mapping.

- **Fix**

    Type: Memo (very large string field).

    Value: Very large number of text characters.

    Internal Identifier: `Fix`.

    Description: Explains the exact fix implemented for the problem recorded in this change request.

    This field maps to the Quality Center field `R & D Comments` and is owned by StarTeam. You can map the field `R & D Comments` to the StarTeam fields `Fix` and `Work Around`. For more information, see "R & D Comments" in *Appendix B - Quality Center Defect Fields* on page 53.

- **Flag**

  Type: Enumerated.

  Value: `No` or `Yes`.

  Internal Identifier: `Flag`.

  Description: Bookmarks or identifies change requests in the change request list. You can flag an item as a reminder to follow up on a customer request. The Flag field displays either `Yes` with a blue flag to indicate that the item has been flagged or `No` to indicate that the item is not flagged.

  Mapping this field may not be useful because it is per-user and, therefore, dependent on the StarTeam logon name in the properties file.

  You can map this field to a Quality Center string field or lookup list field that has the values `No` and `Yes` and that requests verification.

- **Flag User List** (Advanced)

  Value: Byte array displayed as a bracketed series of numbers in hex format. For example, `[14 00 00 00]` indicates a specific user.

  Internal Identifier: `FlagUserList`.

  Description: Internal use only. Identifies users who have set flags on a given item.

  This field should not be mapped.

- **Folder**

  Type: Text.

  Value: Up to 254 characters.

  Internal Identifier: `Folder`.

  Description: Name of the StarTeam folder that stores the change request.

  You can map this field to a Quality Center string field. Folder names are truncated unless their names are restricted to the character limits set in Quality Center for string fields. Errors can be limited by using a list that contains the folder names and that requests verification.

- **Folder Path**

  Type: Text.

  Value: Up to 254 characters.

  Internal Identifier: `Folder Path` (includes a space).

  Description: Path to the StarTeam folder that stores the change request.

  You can map this field to a Quality Center string field. However, folder paths are truncated if they exceed the character limits set in Quality Center for string fields. Mapping this field is not recommended because paths are likely to exceed the Quality Center limits.

- **Last Build Tested**

  Type: Enumerated.

  Value: Either a selection from a list of the build labels in the view or `<None>`.

  Internal Identifier: `LastBuildTested`.

  Description: Build label selected by a user to represent the last build in which a change request was tested.

You can map this field to a Quality Center string field or lookup list field that matches the StarTeam build labels and that requests verification. StarTeam must own this mapping.

- **Modified By**

  Type: User ID.

  Value: Either a selection from a list of users or `<None>`.

  Internal Identifier: `ModifiedUserID`.

  Description: Name of the user who last modified a change request.

  If the Quality Center field uses a list that matches the user logon names in StarTeam and that requests verification, you can map this field to a Quality Center User List field. However, the length of user names in Quality Center is database dependent, while StarTeam allows up to 63 characters. StarTeam must own this mapping.

- **Modifed Time**

  Type: Date/Time.

  Value: Date and time.

  Internal Identifier: `ModifiedTime`.

  Description: Date and time when a change request was last modified.

  If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center date field. However, the time will be lost.

- **New Revision Comment** (Advanced)

  Type: Text.

  Values: text.

  Internal Identifier: `NewRevisionComment`.

  Description: Internal use only. The StarTeam client uses this value during the item update process. If added to the upper pane of the StarTeam client, the field appears empty.

  This field should not be mapped.

- **Object ID** (Advanced)

  Type: Integer.

  Value: Database dependent.

  Internal Identifier: `ID`.

  Description: When added to a view, a change request is assigned an object ID. When a change request is branched in a child view, it is assigned another object ID. The original ID belongs to the change request in the parent view.

  If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center number field.

- **Parent Branch Revision** (Advanced)

  Type: Number.

  Value: `-1`, `0`, or positive integer.

  Internal Identifier: `PathtRevision`.

  Description: The last number in the branch revision number before a change request branched. For example, if this number is 7, the branch revision was 1.7 at the time the change request branched and became

1.7.1.0, as seen in the change request history. If a change request was not inherited from the parent view, This number is -1.

If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center number field.

- **Parent ID** (Advanced)

  Type: Integer.

  Value: Database dependent.

  Internal Identifier: `ParentObjectID`.

  Description: The object ID number of a change request in the parent view. If this view has no parent view, the value in the field `Parent ID` is `-1`.

  If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center number field.

- **Parent Revision** (Advanced)

  Type: Integer.

  Value: Database dependent.

  Internal Identifier: `ParentRevision`.

  Description: Revision number when a change request branched. For example, if this number is 8, the change request revision number in the parent view was 8 when the change request branched. The history will show that revision 9 is the first revision in the current view. If this change request was not inherited from the parent view, this number is 0.

  Iif StarTeam owns the pair of mapped fields, you can map this field to a Quality Center number field.

- **Platform**

  Type: Enumerated.

  Value:

  - `All` (default)
  - `MacOS`
  - `Other`
  - `Unix`
  - `Windows 2000`
  - `Windows 95`
  - `Windows 98`
  - `Windows NT`
  - `Windows XP`

  Internal Identifier: `Platform`.

  Description: Operating system on which the problem occurred.

  The StarTeam field `Platform` cannot be required. However, you can change its name, add values to the enumerated list of operating systems, and change the names of existing values.

  You can map this field to a Quality Center lookup list field that has a list of the values for the field `Platform`.

- **Priority**

  Type: Enumerated.

  Value: `No` or `Yes`.

  Internal Identifier: `Priority`.

Description: The value of the field `Priority` indicates the importance of the problem.

The StarTeam field `Priority` cannot be required. However, you can change its name, add values to the enumerated list, and change the names of existing values. Use repository customization to extend this field to include other values because booleans in StarTeam are treated as enumerated types. For example, `No` is `0` and `Yes` is `1`. An administrator can change `No` to `Not A Priority`, `Yes` to `Priority 1`, and add Priorities 2 through 10.

You can map this field to the Quality Center field `Priority`. The selections in these fields must match. Alternatively, map the StarTeam field `Priority` to a Quality Center lookup list field that has matching values and that requests verification.

- **Read Only** (Advanced)

    Type: Enumerated.

    Value: `No` or `Yes`.

    Internal Identifier: `ReadOnly`.

    Description: Indicates whether the StarTeam change request is read-only. A read-only change request is not floating and cannot branch.

    You can map this field to a Quality Center string field or lookup list field that has the values `No` and `Yes` and that requests verification. StarTeam must own this mapping.

- **Read Status**

    Type: Enumerated.

    Value: `Read` or `Unread`.

    Internal Identifier: `ReadStatus`.

    Description: Indicates whether a change request is read or unread.

    Mapping this field may not be useful because it is per-user and, therefore, depends on the StarTeam logon name in the properties file.

    You can map this field to a Quality Center string field or lookup list field that has the values Read and Unread and that requests verification.

- **Read Status User List**

    Value: Byte array displayed as a bracketed series of numbers in hex format. For example, `[14 00 00 00]` indicates a specific user.

    Internal Identifier: `ReadStatusUserList`.

    Description: Internal use only. Identifies users for whom a given item's status is "unread".

    This field should not be mapped.

- **Resolved On**

    Type: Date/Time.

    Value: Date and time.

    Internal Identifier: `ResolvedOn`.

    Description: Date and time when a change request was resolved. StarTeam automatically completes this field when the status of a change request is resolved. The resolution can be any of the following: `Cannot Reproduce`, `As Designed`, `Fixed`, `Documented`, or `Is Duplicate`.

    If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center date field. However, the time will be lost.

- **Responsibility**

  Type: User ID.

  Value: Either a selection from a list of users or `<None>`.

  Internal Identifier: `Responsibility`.

  Description: Name of the user currently responsible for this change request.

  You can map this field to a custom Quality Center User List field associated with the Users list. However, the Users list must match the StarTeam logon names in length. The length of user names in Quality Center is database dependent, while StarTeam allows up to 63 characters.

- **Revision Flags** (Advanced)

  Value: `0`.

  Internal Identifier: `RevisionFlags`.

  Description: Internal use only.

  This field should not be mapped.

- **Root Object ID** (Advanced)

  Type: Integer.

  Value: Database dependent.

  Internal Identifier: `RootObjectID`.

  Description: `Object ID` of the oldest ancestor of a change request. For example, if a change request was not inherited from a parent view, the `Root Object ID` is the same as its `object ID`. If it was inherited from a parent view, the `Root Object ID` is the `Parent ID`, or the parent's `Parent ID`.

  If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center number field.

- **Severity**

  Type: Enumerated.

  Value: `High`, `Low` (default), or `Medium`.

  Internal Identifier: `Severity`.

  Description: The impact of the problem on the product.

  The StarTeam field `Severity` cannot be required. However, you can change its name, add values to the enumerated list, or change the names of existing values.

  You can map this field either to the Quality Center field `Severity` or to a Quality Center lookup list field with values that match those of the StarTeam field `Severity` and that requests verification.

- **Share State**

  Type: Enumerated

  Value: DerivedShare, Not Shared, or Root Share

  Internal Identifier: ShareState

  Indicates whether this item is shared. Root Share means that the item is shared and is the original (or root) reference. DerivedShare means that the item is shared, but this item is not the original (or root) reference. The meaning of Not Shared is obvious.

  You may want to map this field to a Quality Center string field or lookup list field that has the correct values and requests verification. StarTeam must own this mapping.

- **Short Comment**

    Type: Memo (very large string field)

    Value: The first 2000 characters of the revision comment

    Internal Identifier: ShortComment

    Contains the first 2000 characters of the revision comment, which generally explains the reason for modifying change request properties. (The full text of the revision comment is stored in the Comment field.) Note that modifying the properties causes StarTeam to create a new revision.

    You may want to map this field to a Quality Center string field. However, the data may be truncated when it goes to Quality Center, because the maximum length of Quality Center string fields is database dependent.

- **Status**

    Type: Enumerated.

    Value:

    - `New` (default)
    - `Open`
    - `In Progress`
    - `Deferred`
    - `Cannot Reproduce`
    - `As Designed`
    - `Fixed`
    - `Documented`
    - `Is Duplicate`
    - `Verified Deferred`
    - `Verified Cannot Reproduce`
    - `Verified As Designed`
    - `Verified Fixed`
    - `Verified Documented`
    - `Verified Is Duplicate`
    - `Closed Deferred`
    - `Closed Cannot Reproduce`
    - `Closed As Designed`
    - `Closed Fixed`
    - `Closed Documented`
    - `Closed Is Duplicate`

    Internal Identifier: `Status`

    Description: The state of the change request.

    The StarTeam field `Status` cannot be required. It's enumerated values can be renamed, but no other customization is possible.

    You can map this field to a Quality Center string field or lookup list field that has the correct values and that requests verification.

- **Synopsis**

    Type: Memo (very large string field).

    Value: Very large number of text characters.

Internal Identifier: `Synopsis`.

Description: The summary of the problem.

You can map this field to the Quality Center field `Summary`, which can contain a maximum of 70 characters. As a result, Quality Center must own this field.

- **Test Command**

    Type: Text.

    Value: Up to 254 characters.

    Internal Identifier: `TestCommand`.

    Description: Path to a test program that tests for the problem covered by this change request.

    You can map this field to a string field in Quality Center. However, the maximum length of string fields in Quality Center is database dependent. As a result, this field is generally not mapped because this length is not sufficient for most paths.

- **Type**

    Type: Enumerated.

    Value: `Defect` (default) or `Suggestion`.

    Internal Identifier: `Type`

    Description: The type of change request.

    The StarTeam field `Type` cannot be required. However, you can change its name, add values to the enumerated list, or change the names of existing values.

    You can map this field to a Quality Center string field or lookup list field that has the correct values and that requests verification.

- **Verified On**

    Type: Date/Time.

    Value: Date and time.

    Internal Identifier: `VerifiedOn`.

    Description: Date and time when a change request was verified. The resolution can be any of the following:

    - `Verified Cannot Reproduce`
    - `Verified As Designed`
    - `Verified Fixed`
    - `Verified Documented`
    - `Verified Is Duplicate`

    StarTeam automatically completes this field when the status of a change request becomes verified.

    If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center date field. However, the time will be lost.

- **Version** (Advanced)

    Type: Integer.

    Value: Database dependent.

    Internal Identifier: `RevisionNumber`

Description: The last number in the branch revision number. For example, if the branch revision number is 1.3.1.2, the version is 2.

If StarTeam owns the pair of mapped fields, you can map this field to a Quality Center number field.

- **View**

  Type: Enumerated.

  Value: Either a selection from a list of views or `<None>`.

  Internal Identifier: `ViewID`.

  Description: Name of the view in which the current revision was created.

  This field is not available for mapping.

- **Work Around**

  Type: Memo (very large string field).

  Value: Very large number of text characters.

  Internal Identifier: `WorkAround`

  Description: The description of any workaround available for the problem recorded in this change request.

  This field can be mapped to the Quality Center field `R & D Comments`. If you prefer to map `R & D Comments` to the StarTeam field `Fix`, instruct StarTeam users to ignore `Work Around` and to put both fixes and workarounds in the field `Fix`. Alternatively, you can map `R & D Comments` to both `Fix` and `Work Around`. For more information, see "R & D Comments" in *Appendix B - Quality Center Defect Fields* on page 53.

Appendix B - Quality Center Defect Fields

This section explains the Quality Center defect fields and suggests mappings between these defect fields and the StarTeam change request fields.

Defects are added to Quality Center automatically from test results or manually. When a defect is entered manually, the **New Defect** dialog box appears. The required fields appear in the dialog box in red text.

If a required field is not mapped, the user who opens the defect must enter values in all required fields in order to save changes.

The Quality Center administrator determines which fields are visible, which fields are required, and the order in which the fields appear.

- **Actual Fix Time**

  Type: Number.

  Value: Database dependent.

  Internal Identifier: `BG_ACTUAL_FIX_TIME`.

  Description: Actual number of hours required for the fix. You can map this field to a StarTeam custom number field.

- **Assigned To**

  Type: Lookup List.

  Value: Restricted to the choices in the Users list when verification is requested.

  Internal Identifier: `BG_RESPONSIBLE`.

  Description: Indicates who is currently responsible for the defect.

  It is preferable to use two fields, one for each application, instead of mapping this field to the StarTeam field `Responsibility`. If the two fields are mapped, however, user names must be limited to the maximum length of the Quality Center field, which is database dependent. StarTeam allows 63 characters in user names.

  If history is used to determine the contents of this pair of fields, the workflow associated with the StarTeam field `Status` will affect the field value. StarTeam automatically changes the contents of the field `Responsibility` to the logon name of the user who created the change request: the user whose logon is used by the Synchronizer to access the StarTeam Server. For example, the initial file `BugSync.ini` specifies `Administrator` as the value of `vts_username1` and, therefore, as the user name for logon purposes.

- **Closed in Version**

  Type: String.

  Value: Restricted to the choices in the Version list when verification is requested.

  Internal Identifier: `BG_CLOSING_VERSION`.

  Description: The version in which the defect was closed. You can map this field to a StarTeam custom enumerated field.

- **Closing Date**

Type: Date.

Value: String of characters identifying a date.

Internal Identifier: BG_CLOSING_DATE.

Description: The date when the defect was closed in Quality Center. You can map this field to a StarTeam custom date/time field.

> **Note:** When the value in the StarTeam field Status is Closed, StarTeam automatically sets that date and time as the value of the StarTeam field Closed On.

- **Defect ID**

    Type: Number.

    Value: Database dependent.

    Internal Identifier: BG_BUG_ID.

    Description: Unique number associated with the defect. Quality Center increments this number as new defects are added. The initial properties file BugSynch.ini maps the field Defect ID to the StarTeam field External Reference. Do not change this mapping.

- **Description**

    Type: Memo (very large string field).

    Value: Text that can contain large amounts of data. The size limitations depend upon the type of database being used.

    Internal Identifier: BG_DESCRIPTION.

    Description: A complete description of the problem, usually including the steps to reproduce it.

    It is recommended that you map the Quality Center field Description to the StarTeam field Description, with Quality Center as the owner of the pair. The StarTeam Description field is also a memo field.

- **Detected By**

    Type: User.

    Value: Restricted to the choices in the Users list when verification is requested. These user names should match StarTeam logon names.

    Internal Identifier: BG_DETECTED_BY.

    Description: The name of the user who reported the defect.

    This field cannot be mapped to the StarTeam field EnteredBy because it automatically contains the name of the user who was logged on when the defect was created in StarTeam. That user name comes from the Synchronizer. An example of a user name is "QualityCenter".

- **Detected in Cycle**

    Type: Tree List.

    Value: Restricted to one choice from the Cycle list.

    Internal Identifier: BG_DETECTED_IN_RCYC.

    Description: The release cycle in which the problem was found.

    You can map this field to a StarTeam enumerated field or to a StarTeam text field if it is owned by Quality Center.

- **Detected in Release**

Type: Tree List.

Value: Restricted to one choice from the Release list.

Internal Identifier: `BG_DETECTED_IN_REL`.

Description: The release in which the problem was found.

You can map this field to a StarTeam enumerated field or to a StarTeam text field if it is owned by Quality Center.

- **Detected in Version**

    Type: Lookup List.

    Value: Restricted to the choices in the Version list.

    Internal Identifier: `BG_DETECTION_VERSION`.

    Description: The version or build of the product in which the problem was found.

    You can map this field to the StarTeam field `Last Build Tested` if the Version list contains the names of all StarTeam build labels spelled exactly as they are in StarTeam. This would require significant coordination between the Quality Center and StarTeam administrators because the Quality Center field `Planned Closing Version` uses the same Version list and would need "future" build labels in the list.

- **Detected on Date**

    Type: Date.

    Value: String of characters identifying a date.

    Internal Identifier: `BG_DETECTION_DATE`.

    Description: The date when the defect was reported.

    This field cannot be mapped to the StarTeam fields `Created Time` or `Entered On` because StarTeam completes those fields automatically. It can be mapped to a StarTeam custom date/time field.

- **Estimated Fix Time**

    Type: Number.

    Value: Database dependent.

    Internal Identifier: `BG_ESTIMATED_FIX_TIME`.

    Description: The estimated number of hours required for the fix. You can map this field to a StarTeam custom numerical field.

- **Planned Closing Version**

    Type: Lookup List.

    Value: Restricted to the choices in the Version list.

    Internal Identifier: `BG_PLANNED_CLOSING_VER`.

    Description: The version or build of the product in which this problem should be fixed. You can map this field to a StarTeam custom enumerated field or to a StarTeam text field if it is owned by Quality Center.

    For more information, see "Detected in Version" above.

- **Priority**

    Type: Lookup List.

    Value: Restricted to the choices in the Priority list when verification is requested. You can edit or delete any of these choices, which are the following by default:

- `1-Low`
- `2-Medium`
- `3-High`
- `4-Very High`
- `5-Urgent`

Internal Identifier: `BG_PRIORITY`.

Description: The importance of the defect's fix.

You can map this field to the StarTeam field `Priority`, however the choices in these two fields must be customized to match.

- **Project**

  Type: Lookup List.

  Value: Restricted to the choices in the Project list when verification is requested.

  Internal Identifier: `BG_PROJECT`.

  Description: Defines the product or component where the problem exists.

  You can map this field to the StarTeam field `Component`. The length of the field `Project` is database dependent, while the StarTeam field `Component` can contain 64 characters. Quality Center should own this field to prevent loss of data.

  A StarTeam user unaware of the choices in the Project list may prefer to map this field to a custom, enumerated field.

- **R & D Comments**

  Type: Memo (very large string field).

  Value: Text; can contain large amounts of data.

  Internal Identifier: `BG_DEV_COMMENTS`.

  You can map this field to the StarTeam field `Fix`, the StarTeam field `WorkAround`, or to both fields without loss of data because the Quality Center field `R & D Comments` is a memo field.

  This field should be owned by one of the applications because historical information about this field cannot be stored in Quality Center. For example, the Quality Center field `R & D Comments` can be mapped to the StarTeam field `Fix` with StarTeam as the owner.

- **Reproducible**

  Type: String/Lookup List.

  Value: Selected or not selected.

  Internal Identifier: `BG_REPRODUCIBLE`.

  Description: Indicates whether a problem can be reproduced. You can map this field to a StarTeam text field if it is owned by Quality Center.

- **Run Reference**

  Type: Number.

  Value: Database dependent.

  Internal Identifier: `BG_RUN_REFERENCE`.

  Description: The number of test runs. You can map this field to a StarTeam custom integer field, if appropriate.

- **Severity**

Type: Lookup List.

Value: Restricted to the choices in the Severity list when verification is requested. You can edit or delete any of these choices, which are the following by default:

- `1-Low`
- `2-Medium`
- `3-High`
- `4-Very High`
- `5-Urgent`

Internal Identifier: `BG_SEVERITY`.

Description: The impact of the defect on the project or product.

You can map this field to the StarTeam field `Severity`, but the choices in these two fields must be the same. The Quality Center field `Severity` can be mapped to a StarTeam custom enumerated field or, if Quality Center owns the pair of mapped fields, to a StarTeam text field.

- **Status**

  Type: String/Lookup List.

  Value: Restricted to the choices in the Bug Status list when verification is requested. You can edit or delete any of these choices, which are the following by default:

  - `New`
  - `Open`
  - `Fixed`
  - `Closed`
  - `Reopen`
  - `Rejected`

  Internal Identifier: `BG_STATUS`.

  Description: Indicates the state of the defect, such as Open or Closed.

  The initial properties file maps this field to a StarTeam custom enumerated field, which should have the same states as the Quality Center `Status` field. You can map the Quality Center `Status` field to the StarTeam `Status` field, which has the same states as by default.

  Similarly, a custom Quality Center string field can be associated with a list that matches the StarTeam Status field states. You may prefer to map the Quality Center field Status to an existing, but unused, StarTeam string field because Quality Center must own this mapped pair of fields.

- **Subject**

  Type: Lookup List.

  Value: Restricted to the choices in the Subject list when verification is requested.

  Internal Identifier: `BG_SUBJECT`.

  Description: Usually used to define the category within the product or component where the problem exists.

  The StarTeam text field `Category` has a similar use. However, the length of the Quality Center field `Subject` is database dependent, while the StarTeam field `Category` can contain 64 characters. If you map these two fields, Quality Center should own this field to prevent loss of data.

  A safer approach is to map the Quality Center field `Subject` to a StarTeam custom enumerated field because a developer using StarTeam may not be aware of the choices in the Quality Center Subject list.

- **Summary**

Type: String.

Value: Up to 70 characters.

Internal Identifier: BG_SUMMARY.

Description: A short synopsis of the problem to be resolved. This field should be mapped to the StarTeam field Synopsis. However, Quality Center should own the field Summary because the StarTeam memo field Synopsis can be much larger.

If the field is updated based on its history, only the first 70 characters of information entered in StarTeam are preserved. The rest will be truncated and lost.

- **Target Cycle**

  Type: Tree List.

  Value: Restricted to one choice from the Cycle list.

  Internal Identifier: BG_TARGET_RCYC.

  Description: The release cycle in which the problem will be fixed.

  You can map this field to a StarTeam enumerated field or to a StarTeam text field if it is owned by Quality Center.

- **Target Release**

  Type: Tree List.

  Value: Restricted to one choice from the Release list.

  Internal Identifier: BG_TARGET_REL.

  Description: The release in which the problem will be fixed.

  You can map this field to a StarTeam enumerated field or to a StarTeam text field if it is owned by Quality Center.

- **Test Reference**

  Type: Number.

  Value: Database dependent.

  Internal Identifier: BG_TEST_REFERENCE.

  Description: Number of the test. You can map this field to a StarTeam custom integer field, if appropriate.

- **Test Set**

  Type: String.

  Value: Restricted to the choices in the Test Set list when verification is requested.

  Internal Identifier: BG_CYCLE_REFERENCE.

  Description: Indicates the test set that found or can verify a defect's fix. You can map this field to a StarTeam custom text field. The length of the StarTeam custom text field should be less than or equal to the maximum length for the Quality Center field Test Set, which is database dependent. Quality Center should own this field so that its value really is from the Test Set list.

- **To Mail**

  Type: String field.

  Value: Y or N; This field is restricted to the choices in the Yes No list, and, by default, verification is requested.

  Internal Identifier: BG_TO_MAIL.

Description: Indicates whether information about changes to a defect should be sent to users.

The field `To Mail` is set to `Y` by default, but can be manually changed to `N`. `N` prevents information about changes to a defect from being sent to users When the field contains a `Y`, users are notified about changes based on the notification conditions set up for those users. You can map this field to a StarTeam custom enumerated field or to a StarTeam text field if the mapping is owned by Quality Center.

- **User-defined (custom) fields**

    Type: Number, String, or Date.

    Value: Depends on type, etc.

    Internal Identifier: `BG_USER_01` through `BG_USER_24`.

    Description: These are unused fields that are available for definition by users.

    Note: The Synchronizer does not support any mapping of the following user-defined fields: `BG_USER_HR_01` through `BG_USER_HR_06`.

Related Topics

Appendix C - Initial Synchronizer Properties File

Syntax:

This appendix shows the contents of the `BugSync.ini` file, the sample properties file initially installed with the Synchronizer.

```
# A sample properties file for StarTeam Synchronizer for Quality Center

# The below file defines 4 password directives ( with their optional iteration
modifiers )
# which are encoded in a file having the same name as the .ini file with
# ".passwords.bin" appended.  These directives are:
#    vts_password, mtd_password, mtd_app_password, email_smtp_password
# The user can enter them in the .ini file, run the synchronizer, and then remove
 the directive
# completely ( or commment it out - removing the just the password from the
directive
# implies the password is the empty string ).
# Or, the user can create these directives in a file having the same name as the
 .ini file
# but with ".passwords.txt" appended ( and not define them in the .ini file ).
The
# synchronizer will write the passwords using base64 encoding to a .bin file and
 delete
# the .txt file.  The encoding is meant to provide a minimal level of privacy
regarding
# passwords - the encyrption algorithm could be compromised by decompiling the
synchronizer.
# Users should still protect the .bin file accordingly.

##########################################################################
# This block identifies the projects to be synchronized.  Each field on the left
 is
# optionally indexed with a number by appending an index from 1 to n ( e.g.
# vts_server1=...  The corresponding Quality Center (QC) and StarTeam (ST)
# numbers will be synchronized.
##########################################################################

vts_server=localhost
vts_port=49201
vts_project=StarDraw
# The StarTeam user running the synchronizer must rights to create/update CRs.
# The user must also have the "Administer user accounts" access right
# in order to read the list of user names.  If either of the below options is set
 the
# synchronizer user must have the "Change server security settings" access right
 in order
# to read the access rights to verify sufficient access rights for:
#   If vts_create_custom_fields is enabled, then the ST user must also have
"Add/Modify database schema"
#   If EnteredBy is mapped and owned by TD the ST user must also have "Change
user/operation time"
```

```
vts_username=Administrator
vts_password=Administrator

# If you have more than 1 iteration defined ( e.g. vts_server1, vts_server2, ..
)
# use the below field to indicate the number of iterations.  You only need
# to enter those required fields which are different from the preceeding
# ( lower numbered ) set. If you have only 1 set of fields this field is
# not required. Leaving off the iteration value is equivalent to commenting it
out.
# iterations=1

# Name of ST view containing the CRs to be synchronized.
# Leave view blank to use the project's root (or default) view
vts_view=

# Folder path in ST view where the synchronization should start.
# Leave blank to use the view's root folder. If you are using a child of the root
# folder, use the path to that child folder without the root folder. Use the
forward
# slash (/) as a separator for any other folder names. For example, if the root
# folder of the view is StarDraw and the path to the folder you want to
# synchronize is StarDraw/SourceCode/Client, set vts_folder1 to SourceCode/Client.
vts_folder=

# If you need to use / or \ slash as part of a folder name then you must use the
 replacement directives.
# The value for the directive can not contain / or \, and should not be contained
 in the path for the vts_folder
# directive
# vts_forward_slash_as_name=
# vts_backward_slash_as_name=

# Recurse from the vts_folder through all of its children?
vts_recurse_children=yes

####################################################
# This block defines the JDBC connection used to read data from the TD Database.
# One of the following driver/URL combinations needs to be selected.
# These are not interchangable. If you select the odbc driver you need to
# use the odbc URL. Likewise if you select the Oracle jdbc driver you need to
# select the Oracle jdbc URL.

# The ODBC driver for use with the Demo database installed with TD,
mtd_driver=sun.jdbc.odbc.JdbcOdbcDriver
# The jdbc driver to take advantage of CLOB's in Oracle
#mtd_driver=oracle.jdbc.driver.OracleDriver

# This is the ODBC URL points to the DSN for the Demo database installed with TD,
 if you
# create your own TD database, you will also have to create a DSN for it
# jdbc:odbc:<dsn> where the <dsn> is specified by the ODBC administrator
mtd_dataurl=jdbc:odbc:QualityCenter_Demo
# The jdbc URL should be specified as one of the following
# jdbc:oracle:thin:@<serverName>:<port>:<sid> or
# jdbc:oracle:oci8@<net alias>
#mtd_dataurl=jdbc:oracle:thin:@localhost:1521:ORCL
#mtd_dataurl=jdbc:oracle:oci8:@TDORASERVER

# If the Quality Center Server is running in a different timezone from the
synchronizer
```

```
# specify a timezone id code for where the QC Server is running.
# The list of available codes can be retrieved by specifying
# an invalid code.  Some common codes are US/Pacific, US/Arizona, US/Mountain,
US/Central, US/Eastern
# mtd_timezone=US/Hawaii

# Set the desired codepage, if not set the system will default to the System
codepage.
# The typical windows codepage Windows Latin-1
#mtd_character_encoding=Cp1252
# The codepage for Windows Hebrew
#mtd_character_encoding=Cp1255

# With a Microsoft SQL database, mtd_username may need to be set to "td", the
# default user name that QC assigns to any newly created database.
# Then set mtd_password to "tdtdtd", the password used for this database
# from TD.
mtd_username=td
mtd_password=tdtdtd

# A user with restricted access to the database can be used as the database login
 for the
# synchronizer.  In this case, the table references in the SQL statements
# must have the Quality Center schema user prepended with the
# owner name of the database schema.  The value used is set in the
mtd_table_qualifier directive
# mtd_table_qualifier=schema_owner_name

# For MS SQL the owner is always "td", for Oracle the owner is
<domainName>_<ProjectName>_db

# The sql statements used in the synchronizer will have the form:
#   SELECT * from <mtd_table_qualifier>.BUG

# The permissions assigned to the user identified with the mtd_username directive
 need to only
# include read access to the BUG and SYSTEM_FIELD tables.
# The synchronizer does not use the JDBC connection and credentials for updates.
  It uses the mtd_app credentials
# defined below.

# The synchronizer uses the QC OTA APIs to update the database.  The following 5
 fields
# provide the connection information used by the OTA APIs.

# The mtd_url is the address of the QC Server that you would enter in a browser.
# The mtd_app_domain is the Quality Center domain.  If not specified, DEFAULT is
 used.
# The mtd_app_username and mtd_app_password are the parameters used to log
# into Quality Center. NOTE that these are different from the database logins
specified above.
mtd_url=http://localhost:8080/qcbin
mtd_app_domain=DEFAULT
mtd_app_project=QualityCenter_Demo
mtd_app_username=alex_qc
mtd_app_password=

# The synchronizer can send an email message when a failure occurs during
synchronization.
# You must specify the below parameters.  If sending outside the smtp domain, the
 mail server
```

```
# must support relaying
#email_smtp_host_name=smtp.mybusiness.com
#email_smtp_user=myusername
#email_smtp_password=mypassword
#email_from_address= myusername@mybusiness.com
#email_to_address_list= myboss@mybusiness.com

# Attachments synchronization between QC and StarTeam is enabled by setting either
 the
# copy_attachments_from_td_to_st or COPY_ATTACHMENTS_FROM_ST_TO_TD directive (
or both )
# to YES ( default is NO ).
# Attachments are copied from the source to the target based on the attachment
# name not existing in the target.
# Attachment processing is not necessary to synchronize other fields.
# The OTA APIs are used to read and update attachments in TD.
#copy_attachments_from_td_to_st=Yes
#copy_attachments_from_st_to_td=Yes

# The mtd_project and mtd_criteria directives are used to build the query used
to
# identify which bugs in QC are synchronized.

# The mtd_project directive indicates the value of the BG_PROJECT field in QC for

# the defects to be synchronized.
# Commenting out this field will synchronize all defects in the QC Project
container with ST.
# Setting the value of this field to <All Projects> will also synchronize all
defects.
mtd_project=<All Projects>

# The value of mtd_criteria is a SQL condition which can be used to restrict
# the bugs in QC which are referenced for synchronization.  The user must specify
 a
# valid SQL condition.  It is recommended that the user test his syntax using the
# Site Administrator by typing a test SELECT clause.  For example to constrain
# the synchronizer to bugs which are assigned to james_qc, the sample SELECT would
 be
#   SELECT * FROM BUG WHERE BG_RESPONSIBLE='james_qc'
# In this case the mtd_criteria directive would be:
# mtd_criteria=BG_RESPONSIBLE='james_qc'
# NOTE: Unlike mtd_project, the synchronizer ignores the value of this condition
# when creating new bugs in TD.

############################################################################
# The remaining directives are global and can not be restricted to a particular
# iteration
############################################################################

############################################################################
# The next two properties indicate the names of the fields in StarTeam and
# Quality Center which hold the synchronization flags.  These fields must have
# either "Yes" or "No" as the value.  If the value is other than "Yes" it will
# be treated as "No".
############################################################################

# Name of custom field in ST containing the synchronization flag
# This field must be an enumerated field with values "Yes" and "No".
vts_send_to_mtd=Usr_Sync
```

```
# Name of custom field in QC containing the synchronization flag.
# This field must be a string field that is verified with a list that has the
# values "Yes" and "No".
mtd_send_to_vts=BG_USER_01

# A StarTeam query can also be used as an additional criteria to filter which
bugs in
# StarTeam should be synchronized.  Specify the name of the Query
#vts_query=

# QC memo fields are stored in html.  However StarTeam does not.  By default the

# synchronizer will remove html tags in QC and replace them with tab and crlf
characters
# when mapping to StarTeam.  To have the actual "raw" html content mapped to
StarTeam,
# set the vts_remove_html directive to NO.  The default is "Yes".
#vts_remove_html=NO

# When mapping from StarTeam to QC memo fields, if the data in StarTeam already
looks like
# an HTML field ( it begins with "<html><body>" and ends with "</body></html>" )
 no
# translation will occur.  Otherwise tabs and crlf characters in StarTeam ARE
ALWAYS
# translated into html in TD.  Note that a tab in StarTeam is translated into 8
spaces
# in QC because QC doesn't use the <tab> html tag.

# The vts_use_starteam_user_fullname directive will cause the synchronizer to use
 the fullname
# property rather than the logon name.  The default is no.
#vts_use_starteam_user_fullname=yes

# The following directive specifies whether the synchronizer should automatically
# create custom fields for QC fields that do not currently exist in ST.
# Note: Custom fields in StarTeam always prepend 'Usr_'.  Note: In the 8.0 release
 of the
# synchronizer setting this field to Yes would also add enumerations to existing
 fields
# in StarTeam.  Updating enumerations in both TD and StarTeam is now controlled
via the
# mtd_add_new_enums and vts_add_new_enums directives described immediately below.
#vts_create_custom_fields=Yes

# The following 2 directives control the synchronization of the enumeration/lookup
 lists
# in QC and StarTeam. Customization of the 4 QC fields that use the release and
cycle enumerations
# as well is the BG_SUBJECT field is not supported. The default for both is no
synchronizing.
# They only apply when both the QC field is a lookup list and the
# StarTeam field is an enumeration. LastBuildTested, AddressedIn, and
AddressedInView are
# viewed as enumerations for the purpose of these 2 directives.  Prior to
synchronizing, all
# lists are compared.  Values are added if the owner of a mapping has a value in
 its range
# that isn't in the other side.  For example, given the mapping
BG_USER_06<USR_COLORS,
# if BG_USER_06 is a user defined list ( "red", "green" ) and USR_COLORS is a
```

```
StarTeam enumeration of
# ("red", "green", "blue"), then synchronizer will attempt to add "blue" to the
list used by BG_USER_06.
# The directive can be YES, NO, or a list of QC ( or starteam ) property names.
 The default is NO.

# If the directive does not specify a required value be added to the list, a
warning is generated.

# QC stores its lists in a tree each node having a maximum of 677 children. If
the number of items
# to add would cause the number of children to be greater than ( 677/2 ), the
synchronizer will build
# a hierarchy with the values to add.  ' ' and '.' will are used as delimiters
to find common subelements.
# For example, adding "1.0" and "1.0.1" will cause 1.0.1 to be added as a child
of 1.0.1.
# When synchronizing build labels from StarTeam to TD, this commonly occurs.

# The mtd_add_new_enums and vts_add_new_enums directives can be Yes, No, or a
list of field names
# separated by commas, for which the directive should have a Yes value.  In this
 way, the user
# can select which enum fields will be synchronized.  For example, if BG_USER_06
 is a QC enumeration
# field which is mapped to AddressedIn, then to add new view labels to the
BG_USER_06 enumeration
# and not update any other QC enumerations based on new StarTeam enumeration
values, the directive is:
#    mtd_add_new_enums=BG_USER_06

#mtd_add_new_enums=Yes

# The following directive will cause the synchronizer to add values to StarTeam
enumeration
# properties that are missing when compared to the lookup lists of the mapped QC
 field.
#vts_add_new_enums=Yes

###############################################################################
# When more than 1 Iteration is defined, the synchronizer is able to detect if
# a StarTeam CR is in the folder from one Iteration, but based on its associated
 TD
# properties it should be in a different iteration.  In such cases, the CR
# is moved from is current location in StarTeam to the root folder of the correct
# iteration.
#
# This functionality can have significant memory overhead because all of the CRs
 must be
# collected before any synchronization can be done.  The functionality is on by
# default but can be disabled via:
# VTS_ENABLE_MOVE_CR_CAPABILITY=NO
#
# If the move functionality is required, synchronizer and server memory can be
# held to a minimum ( at the cost of slower performance and server overhead ) by
# keeping only a single StarTeam view open at a time.  This means that during CR
# collection, each view is opened, read, and then closed.  Then during
synchronization
# each view is opened again.  This option by default is on, but can be disabled
# allowing the synchronizer to move CRs in StarTeam faster ... at the expense of
# both client and server memory.  By default, the synchronizer keeps only a single
```

```
# view open at a time.  This option can be disabled by setting the below directive
 to
# NO
# VTS_KEEP_ONLY_ONE_VIEW_OPEN=NO

# If VTS_ENABLE_MOVE_CR_CAPABILITY is set to NO, VTS_KEEP_ONLY_ONE_VIEW_OPEN
# has no effect.  Both of these options have no effect if only a single view is
# referenced ( regardless of the number of Iterations ).
###########################################################################

###########################################################################
# The rest of the file below defines the field mapping used for all of the
# TD/ST projects defined above.
###########################################################################

# Mapping QC defect (Bug) fields into ST change request (CR) fields
# format:   map.QualityCenterDefectFieldName<=>StarTeamCRFieldName
#
#   < means that ST owns this mapping; ST's value for this field becomes
#       the value for both fields.
#
#   = means that the history is used to determine the most recent value
#       and put that value in both fields
#       NOTE: To use =, the QC field must have history support enabled
#
#   > means that QC owns this mapping; QC's value for this field becomes
#       the value for both fields.
#
#
# Field types must match, especially enumerated fields
#
# Do not map the following ST fields unless ST owns the field.
# If some QC field seems to match one of these fields,
# create a ST custom field as a substitute for this ST field
# in the mapping.
#     ChangeNumber, CreatedTime, CreatedUserID
#     and other system generated fields
#
# Do not map the following QC fields unless QC owns the field.
# If some ST field seems to match one of these fields,
# use a QC custom field as a substitute for this QC field
# in the mapping.
#     BG_PROJECT, BG_BUG_ID
#
# If you map a QC field to either the AddressedIn or LastBuildTested fields
# in StarTeam to a QC enumerated field, the synchronizer based on the value
# of mtd_add_new_enums will add any view Labels to the QC enumerated list
# which aren't already defined.  Note, in the 8.0 version of the synchronizer
# adding values to existing enumerations in ST, was done based on the value of
# vts_create_custom_fields.

# Mappings you don't want can be commented out or left out

# The BG_BUG_ID field in Quality Center can be mapped to a StarTeam integer
# or string field.  Quality Center must own this mapping.
map.BG_BUG_ID>ExternalReference

# If BG_BUG_ID is mapped to a StarTeam string field, the user can specify a
# format string to be used to write the Bug_id using the vts_bug_id_format
directive.
# If vts_bug_id_format is not specified, it defaults to TD BUG: %d.  For example,
```

```
# to generate: "Quality Center Bug: 000034" for QC Bug 34, the directive is:
#vts_bug_id_format=Quality Center Bug: %1$06d
# The resultant string should not contain any digits other than those from the
TD Bug number.
# For example, DO NOT USE THE FORMAT: 1st Set of Bugs: %d

# map.BG_STATUS>Usr_TDStatus
# map.BG_RESPONSIBLE=Usr_TDResponsible
# map.BG_PROJECT>Usr_TDProject
# map.BG_SUBJECT>Usr_TDSubject
# map.BG_SUMMARY>Synopsis

# map.BG_DESCRIPTION>Description

# Workaround, Fix and Description are ST memo fields and will be truncated in QC
# unless mapped to QC memo fields. "Fix+Workaround" when owned by ST( "<" ) can
be used
# to combine these two fields.
# map.BG_DEV_COMMENTS<Fix+Workaround

# When mapping to BG_DEV_COMMENTS, (e.g. map.BG_DEV_COMMENTS<COMMENT )
# the entire target field is replaced.  New data is not appended to the field
# as in pressing the "Add Comment" button.

# map.BG_REPRODUCIBLE>Usr_TDReproducible

# Valuemap fields are used to map ST values to QC values.
# Valuemap fields require that a map value exists for that field(see example).
# Ownership of the field will be determined from the map operator and the only
supported
# valuemap operator will be '='. In the case of enumerated fields warnings are
issued
# when the value does not exist.

# NOTE: The below value map references the names of the BG_SEVERITY fields as
they are in QC 9.0
# Earlier versions of QC prefaced the names with #-, e.g. 1-LOW, 2-Medium, 3-High,
 etc

# map.BG_SEVERITY>Severity
# valuemap.BG_SEVERITY.1-Low=Severity.Low
# valuemap.BG_SEVERITY.2-Medium=Severity.Medium
# valuemap.BG_SEVERITY.3-High=Severity.High
# valuemap.BG_SEVERITY.("4-Very High")=Severity.High
# valuemap.BG_SEVERITY.5-Urgent=Severity.High

# map.BG_PRIORITY<Priority

# The StarTeam EnteredBy field can be mapped.  The mapping
# can be < or >.  If the mapping is >, and a StarTeam CR is being created, the
value of the QC field is
# used to log into StarTeam and actually create the CR.  This requires the StarTeam
 user specified in this
# file to have the Change user/operation time access right.  The access right is
 checked prior to any synchronization.
# If the mapping is <, then new CRs are created using the user identified in the
 vts_username directive.
# = is not allowed for the CreatedUserId field.
# If the mapping is > and the value is changed in QC, a warning is generated
# map.BG_DETECTED_BY>EnteredBy
```

```
# map.BG_DETECTION_DATE>Usr_TDDetectionDate
# map.BG_DETECTION_VERSION>Usr_TDDetectionVersion

# map.BG_PLANNED_CLOSING_VER=
# map.BG_ESTIMATED_FIX_TIME=
# map.BG_ACTUAL_FIX_TIME=

# Do NOT map this to the ClosedOn field in ST. It must map to a custom field.
# map.BG_CLOSING_DATE>Usr_TDClosingDate

# map.BG_CLOSING_VERSION=Usr_TDClosingVersion

# map.BG_USER_01<Platform
# Remember not to use the synchronizing value in TD ( in this case BG_USER_02)
for something else!
# map.BG_USER_03<ChangeNumber
# map.BG_USER_04<Status
# map.BG_USER_05<Type
# map.BG_USER_06<AddressedIn
# map.BG_USER_07<LastBuildTested

# map.BG_USER_08<Responsibility
# map.BG_USER_09=
# map.BG_USER_10=
# map.BG_USER_11=
# map.BG_USER_12=
# map.BG_USER_13=
# map.BG_USER_14=
# map.BG_USER_15=
# map.BG_USER_16=
# map.BG_USER_17=
# map.BG_USER_18=
# map.BG_USER_19=
# map.BG_USER_20=
# map.BG_USER_21=
# map.BG_USER_22=
# map.BG_USER_23=
# map.BG_USER_24=

# The Synchronizer does not support any mapping of the following fields:
# map.BG_USER_HR_01
# map.BG_USER_HR_02
# map.BG_USER_HR_03
# map.BG_USER_HR_04
# map.BG_USER_HR_05
# map.BG_USER_HR_06
```

Appendix D - Initial Synchronizer Batch Files

Syntax:

This appendix shows the contents of the Synchronizer batch files `run.bat` and `run-again.bat`.

run.bat (Windows server)

```
rem @echo off

REM Copyright (c) 2012 Borland Software Corporation.  All Rights Reserved.
REM Portions of this product were created using jacoZoom (c)2000, infoZoom.  ALL
 RIGHTS RESERVED

REM - The path to StarTeam installation should be verified.
SET STARTEAM_PATH=C:\Program Files\Borland

REM - The path to StarTeam SDK installation should be verified.
SET STARTEAM_SDK_PATH=%STARTEAM_PATH%\StarTeam SDK 13.0

REM - The path to StarTeam SDK library should be verified.
SET SDK_JAR=%STARTEAM_SDK_PATH%\Lib\StarTeam130.jar

REM - The version of the VM that the SDK is supported on
SET VM_VERSION=Sun1.6.0_29

REM - The path to Oracle client installation should be verified.
SET ORACLE_PATH=C:\oracle\ora92

REM - The path to Oracle JDBC library, this path should be verified.
SET ORACLE_CLASSPATH=%ORACLE_PATH%\jdbc\lib\classes12.zip

REM - The java lib path should be verified.
SET _JAVA_LIB_PATH=%STARTEAM_SDK_PATH%\Lib;%ORACLE_PATH%\bin;.\OTA\bin

REM - The path to the StarTeam SDK installed Java Runtime Environment should be
verified.
SET JAVA="%STARTEAM_PATH%\Java\%VM_VERSION%\bin\java.exe"

SET
_CLASSPATH=mtdsync.jar;spslib.jar;mail.jar;activation.jar;%SDK_JAR%;%ORACLE_CLASSPATH%;.\OTA\lib\izmcomjni.jar;.\OTA\lib\IDAPI01ELib92.jar

REM - make a call to see which APIs are loaded
%JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"
com.starbase.mtdsync.App APITest

if not errorlevel 1 goto APIOK
SET
_CLASSPATH=mtdsync.jar;spslib.jar;mail.jar;activation.jar;%SDK_JAR%;%ORACLE_CLASSPATH%;.\OTA\lib\izmcomjni.jar;.\OTA\lib\IDAPI01ELib90.jar
:APIOK

REM - Call the synchronizer
%JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"
com.starbase.mtdsync.App BugSync.ini
```

```
REM - The synchronizer has the below options which can appear at the end of the
execution line.
REM -    -help    prints the below list
REM -    log <directory> specify alternate directory for the log file
REM -    debug          generate a more detailed log file
REN -    briefdebug  generate some log messages, but not as much as debug mode
REM -    nobuffer        turn off buffering and enable realtime logging
REM -    readonly        run synchronizer without updating any data
REM -    repeat n         loop every n seconds until Enter is pressed in the cmd
window
REM -    postSyncBat <filePath>  used with repeat, path to .bat file to execute
after each sync

REM - If you want to have the mtdsync.log and mtdsync.err files generated in a
directory other than the
REM -    current directory use the log <dir> option:
REM %JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"
com.starbase.mtdsync.App BugSync.ini log "c:\inetpub\wwwroot\tdlogs"

REM - The syntax below shows how to set the debug flag to generate a more detailed
 mtdsync.log file
REM %JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"
com.starbase.mtdsync.App BugSync.ini debug

REM - The syntax below shows how to set the briefdebug flag which an abbreviated
 debug setting
REM %JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"
com.starbase.mtdsync.App BugSync.ini briefdebug

REM - The syntax below shows how to set the nobuffer to force logging out be
immediately written - this slows down the synchronizer
REM %JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"
com.starbase.mtdsync.App BugSync.ini nobuffer

REM - The syntax below shows how to set the readonly flag to test the synchronizer
 without updating either StarTeam or Mercury data
REM %JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"
com.starbase.mtdsync.App BugSync.ini readonly

REM - The syntax below shows how to use the repeat option and to call a batch
file after each synchronization
REM %JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"
com.starbase.mtdsync.App BugSync.ini repeat 5 postSyncBat "C:\Program
Files\Borland\StarTeam QualityCenter Synchronizer\postSync.bat"

if errorlevel 1 goto Failed
goto Done

:Failed
    echo. Errors occurred. Check mtdsync.log and/or mtdsync.err for further
information.
    goto Bye
:Done
    echo Done. Check mtdsync.log for further information.
:Bye
```

run-again.bat (Windows server)

```
@echo off

REM modified run.bat file to wait some number of seconds between synchronize calls
```

```
REM Copyright (c) 2012 Borland Software Corporation.  All Rights Reserved.
REM Portions of this product were created using jacoZoom (c)2000, infoZoom.  ALL
 RIGHTS RESERVED

REM - edit the parameter to wait.exe at the bottom of this file
REM - type ^C to exit the batch file

echo This process may take some time, please wait...

REM - The path to StarTeam installation should be verified.
SET STARTEAM_PATH=C:\Program Files\Borland

REM - The path to StarTeam SDK installation should be verified.
SET STARTEAM_SDK_PATH=%STARTEAM_PATH%\StarTeam SDK 13.0

REM - The path to StarTeam SDK library should be verified.
SET SDK_JAR=%STARTEAM_SDK_PATH%\Lib\StarTeam130.jar

REM - The version of the VM that the SDK is supported on
SET VM_VERSION=Sun1.6.0_29

REM - The path to Oracle client installation should be verified.
SET ORACLE_PATH=C:\oracle\ora92

REM - The path to Oracle JDBC library, this path should be verified.
SET ORACLE_CLASSPATH=%ORACLE_PATH%\jdbc\lib\classes12.zip

REM - The java lib path should be verified.
SET _JAVA_LIB_PATH=%STARTEAM_SDK_PATH%\Lib;%ORACLE_PATH%\bin;.\OTA\bin

REM - The path to the StarTeam SDK installed Java Runtime Environment should be
verified.
SET JAVA="%STARTEAM_PATH%\Java\%VM_VERSION%\bin\java.exe"

SET
_CLASSPATH=mtdsync.jar;spslib.jar;mail.jar;activation.jar;%SDK_JAR%;%ORACLE_CLASSPATH%;.\OTA\lib\izmcomjni.jar;.\OTA\lib\IDAPIOLELib92.jar

REM - make a call to see which APIs are loaded
%JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"
com.starbase.mtdsync.App APITest

if not errorlevel 1 goto APIOK
SET
_CLASSPATH=mtdsync.jar;spslib.jar;mail.jar;activation.jar;%SDK_JAR%;%ORACLE_CLASSPATH%;.\OTA\lib\izmcomjni.jar;.\OTA\lib\IDAPIOLELib90.jar
:APIOK

:again

REM - Call the synchronizer
%JAVA% -Djava.library.path="%_JAVA_LIB_PATH%" -classpath "%_CLASSPATH%"
com.starbase.mtdsync.App BugSync.ini

if errorlevel 1 goto Failed
if exist mtdsync.err goto Failed
goto Done

:Failed
    type mtdsync.log >> all.log
    type mtdsync.err >> all.err
    echo. Errors occurred. Check mtdsync.log and/or mtdsync.err for further
```

```
information.
    goto Bye
:Done
    echo Done. Check mtdsync.log for further information.
:Bye

REM The parameter to wait is the number of seconds.
    wait 600
    goto again
```

Related Topics

Index

install 7
integers in StarTeam 22
iterations
    properties file directive 13

L

Last Build Tested
    StarTeam Field 45
lookup lists in Quality Center 22

M

mapping
    restrictions 27
mapping to text fields 24
mappings (required) 27
Modified By
    StarTeam Field 46
Modified Time
    StarTeam Field 46
modifying run.bat 32
mtd_add_new_enums
    properties file directive 23
mtd_app_domain
    properties file directive 17
mtd_app_password
    properties file directive 17
mtd_app_project
    properties file directive 17
mtd_app_username
    properties file directive 17
mtd_criteria
    properties file directive 17
mtd_dataurl
    properties file directive 17
mtd_driver
    properties file directive 18
mtd_password
    properties file directive 18
mtd_project
    properties file directive 18
mtd_send_to_vts
    properties file directive 18
mtd_table_qualifier
    properties file directive 18
mtd_url
    properties file directive 19
mtd_username
    properties file directive 19

N

New Revision Comment
    StarTeam Field 46
numbers
    in StarTeam 22
numbers in Quality Center 22

O

Object ID
    StarTeam Field 46
overview 7
ownership (determined by mapping directives) 27

P

Parent Branch Revision
    StarTeam Field 46
Parent ID
    StarTeam Field 47
Parent Revision
    StarTeam Field 47
Planned Closing Version
    Quality Center field 55
Platform
    StarTeam Field 47
Priority
    Quality Center field 55
    StarTeam Field 47
product support 5
Project
    Quality Center field 56
properties files
    .ini files 27
    mapping 27
    sample 60

R

R & D Comments
    Quality Center field 56
Read Only
    StarTeam Field 48
Read Status
    StarTeam Field 48
Read Status User List
    StarTeam Field 48
real values
    mapping 27
receiving failure email 19
Reproducible
    Quality Center field 56
Resolved On
    StarTeam Field 48
Responsibility
    StarTeam Field 49
restricted mappings 27
Revision Flags
    StarTeam Field 49
Root Object ID
    StarTeam Field 49
Run Reference
    Quality Center field 56
run.bat 32

S

setting up Quality Center for use with StarTeam 8
setting up StarTeam for use with Quality Center 8
Severity
    Quality Center field 56
    StarTeam Field 49
Share State
    StarTeam Field 49
Short Comment
    StarTeam Field 50
Status
    Quality Center field 57
    StarTeam Field 50
strings in Quality Center 22
Subject
    Quality Center field 57
Summary
    Quality Center field 57
support 5
synchronization flags 36
Synopsisi
    StarTeam Field 50
system requirements 6

T

Target Cycle
    Quality Center field 58
Target Release
    Quality Center field 58
technical support 5
Test Command
    StarTeam Field 51
Test Reference
    Quality Center field 58
Test Set
    Quality Center field 58
text in StarTeam 22
To Mail
    Quality Center field 58
troubleshooting 36
truncating
    fields 37
Type
    StarTeam Field 51
types
    Quality Center dates 22
    Quality Center lookup lists 22
    Quality Center numbers 22
    Quality Center strings 22

types *(continued)*
    Quality Center user lists 22
    real numbers in StarTeam 22
    StarTeam dates 22
    StarTeam enumerations 22
    StarTeam integers 22
    StarTeam numbers 22
    StarTeam text 22
    StarTeam user IDs 22
    times in StarTeam 22

U

user IDs in StarTeam 22
user lists in Quality Center 22
User-defined (custom) fields
    Quality Center 59

V

Verified On
    StarTeam Field 51
Version
    StarTeam Field 51
View
    StarTeam Field 52
views
    read-only 37
vts_add_new_enums
    properties file directive 23
vts_bug_id
    properties file directive 27
vts_create_custom_fields
    properties file directive; 21
vts_folder 15
vts_password 14
vts_port 13
vts_project 13
vts_query 16
vts_recurse_children 15
vts_send_to_mtd 16
vts_server 13
vts_use_starteam_user_fullname 16
vts_username 14
vts_view 15

W

Work Around
    StarTeam Field 52