HelixCore

Helix Defect Tracking Gateway Guide

2018.1

August 2019

PERFORCE

www.perforce.com

Contents

How to use this guide

This guide tells you how to install and configure P4DTG, the Helix Defect Tracking Gateway. It is intended for anyone using P4DTG to replicate data between Helix Core Server and a defect tracker. For details about this release, refer to the P4DTG *Release Notes*.

Feedback

How can we improve this manual? Email us at manual@perforce.com.

Other documentation

See https://www.perforce.com/support/self-service-resources/documentation.

Syntax conventions

Helix documentation uses the following syntax conventions to describe command line syntax.

Notation	Meaning
`literal`	Must be used in the command exactly as shown.
italics	A parameter for which you must supply specific information. For example, for a *serverid* parameter, supply the ID of the server.
[`-f`]	The enclosed elements are optional. Omit the brackets when you compose the command.
...	■ Repeats as much as needed: • `alias-name[[$(arg1)... [$(argn)]]=transformation` ■ Recursive for all directory levels: • `clone perforce:1666 //depot/main/p4... ~/local-repos/main` • `p4 repos -e //gra.../rep...`
element1 | *element2*	Either *element1* or *element2* is required.

P4DTG overview

Helix Core, also referred to as Helix Core Server or Helix Server, includes a basic issue-tracking feature called jobs. Jobs can be associated with changelists, so that, for example, when you check in the files that fix a bug or implement a new feature, the associated job is closed.

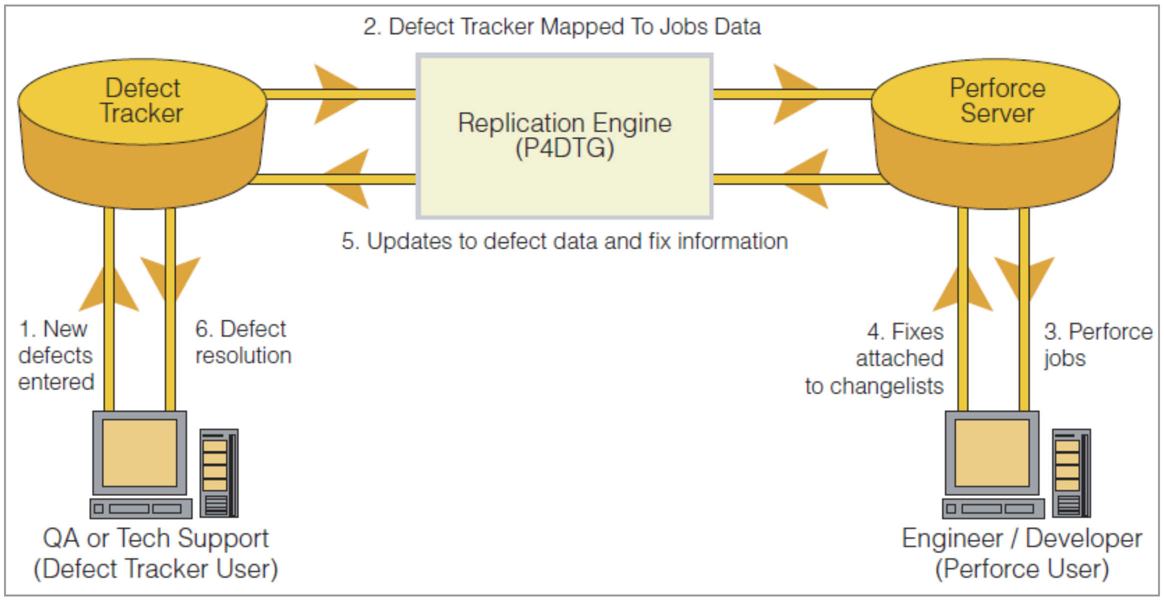
P4DTG enables you to integrate Helix Serverjobs with external defect trackers so that changes to the status of defect tracker issues are replicated in Helix Server jobs and vice versa. For example, when a programmer checks code into Helix Server, thereby fixing a job, the fix information can be propagated to your defect tracker to notify QA engineers that the fix is ready for testing. The gateway enables you to specify how data is propagated between the two systems and provides a program that replicates data according to your specification.

Perforce offers a variety of plug-ins to support various defect trackers, including HP Quality Center, Bugzilla, JIRA, and Redmine. For an up-to-date list of supported defect trackers, consult the Perforce website. After you install a plug-in, be sure to read the product-specific documentation that is installed in the `p4dtg/doc/` directory.

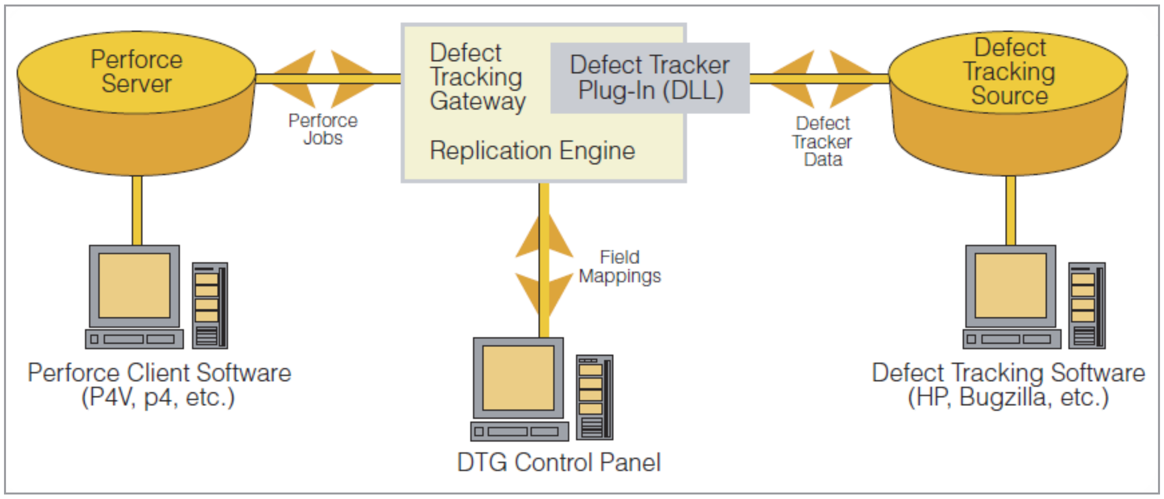
Important
Defect replication is one-to-one, meaning that for each new defect, one Helix Server job is created. You cannot configure P4DTG to map multiple jobs to a single defect, or multiple defects to a single job.

The following figure shows the flow of data between the defect tracker and Helix Server, which follows this sequence:

1. QA or Support engineer enters a defect into the defect tracker.
2. P4DTG replicates the defect in Helix Server as a job.
3. A software engineer reads the Helix Server job and fixes the defect.
4. The software engineer attached the job to a changelist containing the fixed files and checks in the changelist, fixing the job in Helix Server.
5. P4DTG replicates the fix from Helix Server to the defect tracker.
6. The QA or Support engineer sees the fix in the defect tracker.


2. Defect Tracker Mapped To Jobs Data
Defect Tracker
Replication Engine (P4DTG)
Perforce Server
5. Updates to defect data and fix information
1. New defects entered
6. Defect resolution
4. Fixes attached to changelists
3. Perforce jobs
QA or Tech Support (Defect Tracker User)
Engineer / Developer (Perforce User)

The following diagram provides an overview of the gateway components and shows how data is transferred.



The data stored for each job is defined by the Perforce job specification, which can be modified by a user with administrative privileges. For details about modifying the job specification, refer to *Helix Core Server Administrator Guide: Fundamentals*.

By default, Helix Server does not maintain a history of changes to jobs. However, you can create a spec depot in which all changes to Helix Server specifications are archived. For details about spec depots, refer to *Helix Core Server Administrator Guide: Fundamentals*.

For each type of defect tracker (for example, HP Quality Center), a plug-in is required. The plug-in controls which defect tracker data is made available to the gateway. To propagate data between Helix Server and defect trackers, you map Helix Server job fields to defect tracker fields and specify how they are propagated.

To integrate your defect tracker with Helix Server, you must:

1. Install the gateway.
2. Add required fields to the Helix Server job specification.
3. Configure the gateway.
4. Start the replication engine.

For details, see "Installing and configuring P4DTG" on page 8.

Installing and configuring P4DTG

This chapter provides instructions on installing and configuring P4DTG.

Installing P4DTG

To install P4DTG on Unix, download the appropriate file for your platform from the Perforce website. Windows users: run the installer. Unix and Linux users: unpack the archive file.

The following files and folders are installed:

Directory	Contents
`p4dtg/`	Executables for the replication engine, configuration control panel, test tool, and any required libraries
`p4dtg/config`	XML files containing field mappings, source definitions and replication settings
`p4dtg/help`	Online help for configuration control panel
`p4dtg/repl`	Status and error logs for the replication engine

To replicate with the HP Quality Center defect tracker, you must install the HP Quality Center connectivity add-in, a COM-based DLL that is used by the Quality Center plugin to communicate with the Quality Center server. Note that the Quality Center plugin is available only for Windows.

To install the HP Quality Center connectivity add-in:

1. Launch Internet Explorer.
2. Go to the start page of your HP Quality Center installation.
3. Click **Add-Ins Page**. The **Add-Ins** page is displayed.
4. Click **HP Quality Center Connectivity**. The **Add-Ins** page is displayed.
5. Click **Download Add-in**, and download and save `TDConnect.exe`.
6. Run `TDConnect.exe`.

Adding required fields to the Helix Server job specification

To edit the Helix Server job specification, you must have Helix Server admin privilege. To add the required fields:

1. At the command line, issue the `p4 jobspec` command. Helix Server launches a text editor displaying the job specification.
2. Add the following fields (`nnn` indicates field numbers, which depend on what's already defined in your job specification. For details, refer to the description of the `p4 jobspec` command in the *P4 Command Reference*.)

   Note
   The DTG_MAPID field is required only if you intend to define segments for filtered replication. For details, see "Filtered replication" on page 15.

   ```
   nnn DTG_FIXES text 0 optional
   nnn DTG_DTISSUE word 32 optional
   nnn DTG_ERROR text 0 optional
   nnn DTG_MAPID word 32 optional
   ```
3. If not already defined, add a date field that stores the date when the job was last modified and a word field that contains the name of the user who last modified the job. In order to guarantee that the data is propagated, the field-type attribute should read always. For example:

   ```
   nnn ModDate date 20 always
   nnn ModBy word 32 always
   ```
   (The preceding field names are examples. You can assign any valid field name when you edit the job specification.)
4. Define the default value for the preceding fields as follows:

   ```
   ModDate $now
   ModBy $user
   ```
5. Exit the editor, saving your changes.

The attributes you assign to fields in the Helix Server job specification determine how the fields can be replicated in the defect tracker, as follows:

Job specification attribute	Valid replication methods
`always`	Copy to defect tracker
`once`	Copy to defect tracker
`optional`	Copy to defect tracker or mirror in defect tracker

Job specification attribute	Valid replication methods
`required`	Copy to defect tracker or mirror in defect tracker

Configuring P4DTG

To specify how data is replicated between Helix Server and the defect tracker, you use the P4DTG configuration control panel. To launch the configuration control panel, double-click `p4dtg-config.exe`. To configure replication, perform the following steps:

1. Define defect tracking sources
2. Define Helix Server sources
3. Map job fields to defect fields, optionally including replication of fixes

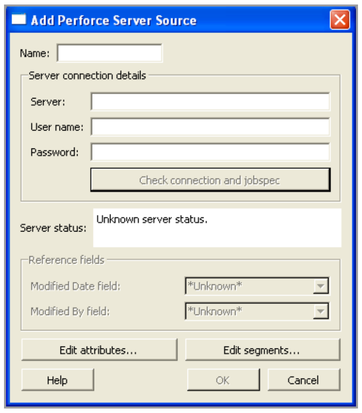
Note
The data you enter, change, or delete using the P4DTG configuration control panel is stored (to configuration files on disk) when you click **Apply** or **OK** on one of the main tabs. Clicking **OK** on the data entry dialogs stores your entries temporarily, but they are not stored permanently until you click **Apply** or **OK** on one of the main tabs.

Defining defect tracker sources

For defect trackers, the connection information depends on the requirements of the defect tracker server. For HP Quality Center, the connection information is specified using the syntax `host:port`. In addition to specifying the connection information, some defect trackers (such as HP Quality Center) require you to specify the project to which you are mapping Helix Server jobs.

To add a defect tracking source:

1. Click **New**. The **Add Defect Tracking Source** dialog is displayed, as shown in the following figure.



Server status: Unknown server status.

Modified Date field: *Unknown*

Modified By field: *Unknown*

2. Enter the required information as follows:

Field	Description
Name	An alias for the source and its associated information. Specify the defect tracker type by choosing the type from the drop-down list (if you have installed plug-ins for multiple defect trackers). To connect to a Jira Software Cloud server, select **JIRA-REST** from the drop-down list.
Server	The information required by the defect tracker for client connections. The format is determined by the defect tracker, for example: ▪ For HP Quality Center, specify `host:port`. ▪ For Jira Software Cloud, specify your Jira Software Cloud server: `https://<your-site-name>.atlassian.net`.
User name	The user name to be used to connect to the defect tracking server. For Jira Software Cloud, `<your-atlassian-account>`.
Password	The password for the specified user, if required. **Jira Software Cloud:** the password is a special API token. To obtain your API token: a. Go to https://id.atlassian.com/manange/api-tokens b. Log in with your Atlassian administrator credentials. c. Click **Create** and name your API token something recognizable, for example `p4jira`. d. The API token is displayed in a dialog. e. Copy the token and paste it into the **Password** field. It is a good idea to save the API token somewhere safe in case you need it in the future.
Reference Field	The fields that are required to keep Helix Server and the defect tracker synchronized.
Modified Date Field	The date that the issue was last changed.
Modified By Field	The user that last changed the issue (if supported by your defect tracker).

The plugin for your defect tracker might enable you to specify other proprietary settings; if so, click **Edit Attributes...** to change these settings. For details, see "Configuring plug-in attributes" on the facing page.

If you intend to use filtered replication for this data source, you must define segments. To define segments, click **Edit Segments...**. For details, see "Filtered replication" on page 15.

3. Click **Check connection and retrieve fields**. If you have correctly specified valid values, the dialog displays a success message and the reference fields retrieved from the defect tracker.

   If a connection error is displayed, correct your entries and click **Check connection and retrieve fields** again.

4. Choose the defect tracker project from the **Project** drop-down list.
5. Click **OK** to save your entries and dismiss the dialog.

Your newly-defined defect tracking source is now displayed in **Defined Defect Tracking Sources**.

To edit a defect tracking source, click the source in the **Defined Defect Tracking Sources** list, then click **Edit**. To delete a defect tracking source, click the source in the **Defined Defect Tracking Sources** list, then click **Delete**. The source is marked for deletion, as indicated in the **Defined Defect Tracking Sources** list. To finish deletion, click **OK** or **Apply**. To undo deletion, click the source marked for deletion, then click **Undelete**.

Configuring plug-in attributes

Specific defect trackers might require settings outside of those required to configure replication. For example, Bugzilla enables you to specify a MySQL database other than its default database. To configure settings for specific plug-ins, edit its attributes.

To edit attributes, click **Edit Attributes...**. For details about a specific field, click its help button (which is labeled with a question mark). Attributes and valid entries are specific to each plug-in, and your defect tracker might not require you to set attributes.

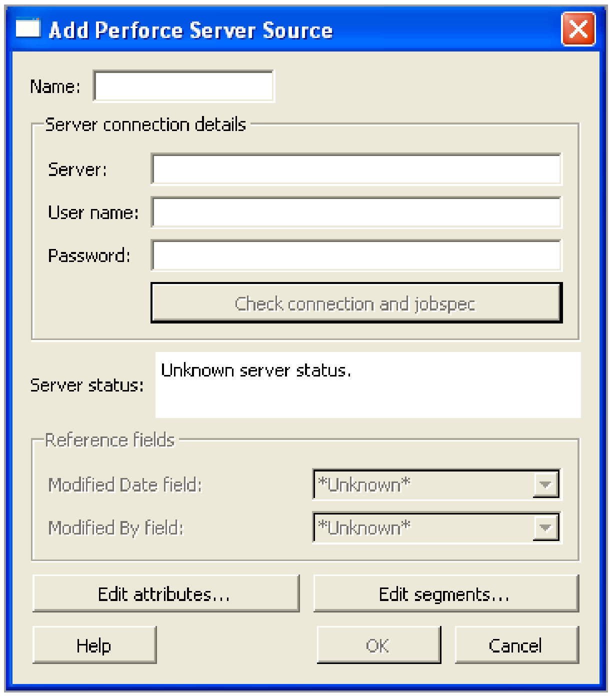
Default attribute settings, which are provided by the plug-in, are displayed in light gray. Required attributes are marked with an asterisk (*). Attribute settings are validated by the plug-in when you click **OK**.

Defining Helix Server sources

To define Helix Server sources, you specify the information required to connect to the server (the host machine and TCP/IP port) using the syntax `host:port`. The default host is `localhost`, and the default port is `1666`.

To add a Helix Server source:

1. Click **New**. The **Add Perforce Server Source** dialog is displayed, as shown in the following figure.



Unknown server status.

Modified Date field: *Unknown*

Modified By field: *Unknown*

2. Enter the server connection details as follows:

Field	Description
Name	An alias for the Helix Server and its associated connection information.
Server	The name of the Helix Server host machine and the port on which it is running. Specify as `host:port`; for example `4host 1667`. The server host defaults to `localhost` and the port defaults to `1666`.
User name	The Helix Server user name used by the replication engine to connect to the Helix Server. **Important** To ensure that you can distinguish jobs modified by the gateway, create a dedicated Helix Server user to be used only by the gateway. The user name must be used only for defect replication. Do not use this user name for any other purpose. Ensure that the user name is neither a prefix nor a suffix of any other user name. For example, if you have a user named **`admin-logger`**, do not use **`admin`** or **`logger`** as the user name.
Password	The password for the specified user, if required.

   If you intend to use filtered replication for Helix Server jobs, you must define segments. To define segments, click **Edit Segments...**. For details, see "Filtered replication" on the next page.

3. Click **Check connection and jobspec**. If you have entered valid values, configuration control panel retrieves the job specification for the specified server and displays the names of the reference fields, which are the Helix Server job data fields required for replication.

   If a connection error is displayed, correct your entries and click **Check connection and jobspec** again.

   If the **Server status** field says that the server is not configured, you must add the required fields to the Helix Server job specification. For details, see "Adding required fields to the Helix Server job specification" on page 9.

4. Click **OK** to save your entries and dismiss the dialog.

Your newly defined Helix Server source is now displayed in **Helix Server sources**.

Helix Server sources that have been edited and need to be saved are displayed in bold.

To edit a Helix Server source, click the source in the Helix Server sources list, then click **Edit**. To delete a Helix Server source, click the source in the Helix Server sources list, then click **Delete**. The source is displayed in bold, marked for deletion, but is not deleted until you click **OK** or **Apply**. To undo deletion, click the marked source, then click **Undelete**.

Filtered replication

To replicate sets of selected defects from a single data source to multiple other defect trackers or version control systems, you define *segments*. For example, if you use Bugzilla to track your database bugs and HP Quality Center to track your Web application bugs, you can create separate segments for each project and replicate them independently. You can create a segment named “Database” that selects all Helix Server jobs with the **Subsystem** field set to “db” and another segment named “Web Application” that selects all Helix Server jobs with the **Subsystem** field set to “webapp.”

To use segments for replication, choose the segment when defining mappings on the **Gateway Mappings** tab. Note the following requirements:

- If you segment a Helix Server source or a Defect Tracking Source, you are required to choose a segment and cannot choose the top-level source.
- A non-segmented data source that is in use (that is, already referenced by a mapping) cannot be segmented.
- After a data source has been segmented and one or more of the segments has been mapped, you cannot remove SELECT values from the segment definition.
- If a data source is already being replicated, you cannot create segments for it.
- Fields only appear in the Helix Server Source segmentation list if they are defined as SELECT fields in the Helix Server job specification.

To create a segment for a Helix Server or Defect Tracking Source:

1. On the **Helix Server Source** or **Defect Tracking Source** tab, click **Edit Segments...**. The **Edit Segments** dialog is displayed.
2. Specify the field to be used to filter defects by choosing the field from the **Segment using SELECT field** list. (The fields listed are SELECT fields returned by the data source plugin.)
3. To add a segment, click **New...**. In the **New Segment** dialog, specify a name for the segment and click **OK**.
4. Specify the values that you want to include in the segment by moving them from the **Unassigned options** column to the **Included options** column.
5. To save your segment definition and dismiss the dialog, click **OK**.
6. To save your changes, click **OK** on the **Helix Server Source** or **Defect Tracking Source** tab.

Mapping data between Helix Server and P4DTG

After defining Helix Server sources and defect tracker sources, you must specify how data is replicated between them when a job or issue is changed. In addition, you can configure the data that is transmitted to the defect tracker when a job is fixed in Helix Server.

Before entering mappings, define the relationships between the data in Helix Server jobs and the data stored by your defect tracker. To display the fields defined for Helix Server jobs, issue the `p4 jobspec` command. (For details about modifying the job specification, see the *Helix Core Server Administrator Guide: Fundamentals*.) For details about the data that is stored by your defect tracker, consult the documentation supplied by the defect tracker vendor.

Fields can be replicated in two ways:

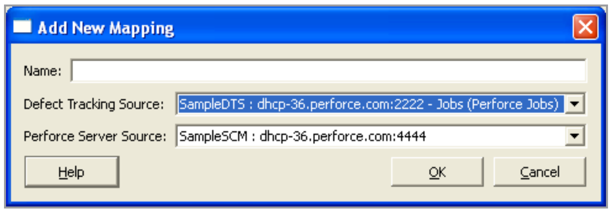
- Copy: any change in the value is sent from the source system to the target system (one-way replication)
- Mirror: any change in the value in either system is replicated in the other system (two-way replicion)

Every field has a data type, and mappings between fields must ensure that the data types are compatible, as described in the following table.

Field	Description	Valid target data types	
		Copy	Mirror
word	A one-word entry	**word**, **line** or **text**	**word**
line	A single line of text	**line** or **text** For targets of type **word**, only the first word is copied.	**line**
text	Multiple lines of text	**text** or **line** For targets of type **line**, only the first line is copied. For targets of type **word**, only the first word is copied.	**text**
date	A date and time	date	**date**
select	A field that is assigned one of a specified set of values. The number of possible values much be identical for Helix Server and the defect tracker.	**select**, **word**, **line**, **text** For **select** fields, all possible values must be mapped.	**select** All possible values must be mapped.

To add a set of mappings:

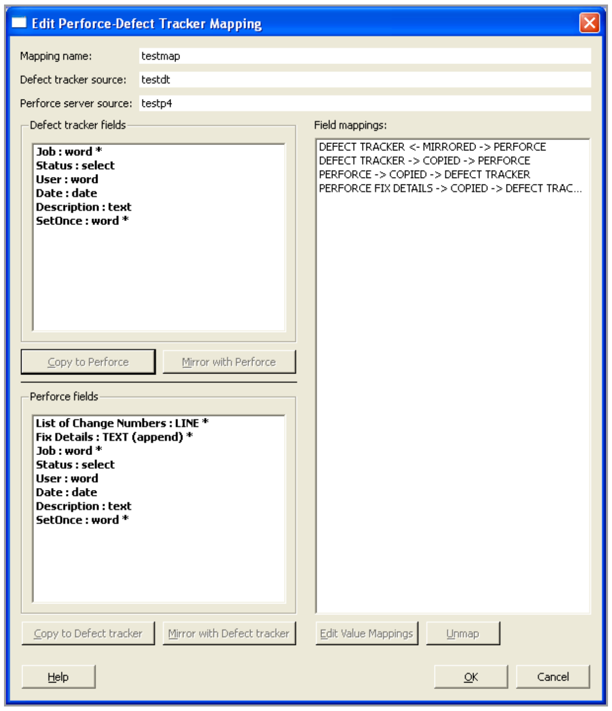
1. On the **Mappings** tab, click **New**. The **Add New Mapping** dialog is displayed.



Name:
Defect Tracking Source: SampleDTS : dhcp-36.perforce.com:2222 - Jobs (Perforce Jobs)
Perforce Server Source: SampleSCM : dhcp-36.perforce.com:4444

2. Enter a name for the mapping, and choose the Helix Server source and the defect tracking source that you want to map. Click **OK**. The **Edit Perforce-Defect Tracker Mapping** dialog is displayed as shown in the following figure.

   List of Change Numbers : LINE * and **Fix Details : TEXT (append)** * are the fields used for replicating Helix Server fix information.



Mapping name: testmap
Defect tracker source: testdt
Perforce server source: testp4

Defect tracker fields:
Job : word *
Status : select
User : word
Date : date
Description : text
SetOnce : word *

Field mappings:
DEFECT TRACKER <- MIRRORED -> PERFORCE
DEFECT TRACKER -> COPIED -> PERFORCE
PERFORCE -> COPIED -> DEFECT TRACKER
PERFORCE FIX DETAILS -> COPIED -> DEFECT TRAC...

Perforce fields:
List of Change Numbers : LINE *
Fix Details : TEXT (append) *
Job : word *
Status : select
User : word
Date : date
Description : text
SetOnce : word *

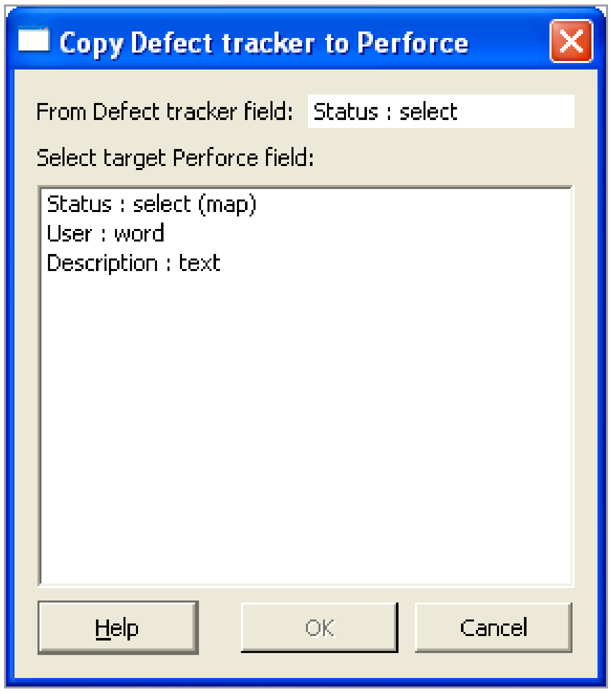
The **Edit Mapping** dialog contains the following fields:

- **Defect tracker fields** lists fields defined in the defect tracker that you can choose to replicate in Helix Server jobs. Unmapped fields are displayed in bold.
- **Perforce** fields lists fields from the Helix Server job specification that you can choose to replicate in the defect tracker. Unmapped fields are displayed in bold.
- **Field mappings** lists the mappings you have defined, categorized by the method and direction of replication. The mappings are displayed using a tree format, grouped by type.

Read-only fields are indicated by an asterisk.

3. For each field you want to replicate, select the field and choose the method of replication, as follows:
   - To replicate **Perforce fields** in the defect tracker, select the source field in the **Perforce fields** pane and click the **Copy to Defect tracker** or **Mirror with Defect tracker** button.
   - To replicate **defect tracker fields** in Helix Server, select the source field in the **Defect tracker fields** pane and click the C**opy to Perforce** or **Mirror with Perforce** button.

   When you click a **Copy** or **Mirror** button, a list of available target fields is displayed, as shown in the following diagram.

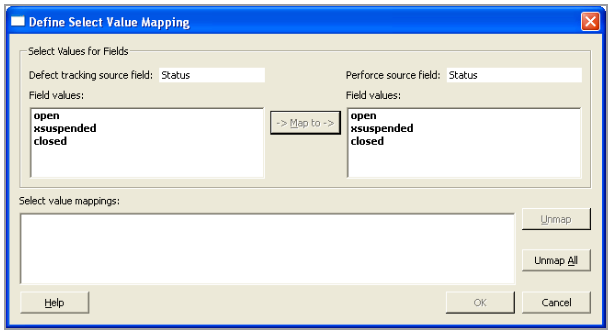
   From Defect tracker field: Status : select

   Select target Perforce field:

   Status : select (map)
   User : word
   Description : text

   If no valid target fields exist, the list is empty.
4. Choose the target field to which you want to map the selected source field and click **OK**.

5. If the target field is a select field (meaning a field that can contain one of a defined set of values), the **Define Select Value Mapping** dialog is displayed, as shown in the following figure.



Defect tracking source field: Status

Field values: open, xsuspended, closed

Perforce source field: Status

Field values: open, xsuspended, closed

Bold type indicates unmapped fields.

6. Specify the correspondence between values in Helix Server and those in the defect tracking system by selecting the Helix Servervalue and its corresponding defect tracker value and clicking **Map to**. Your mappings are listed in the **Current Select Value Mappings** field.

   All the values of the left-hand list must be mapped. You can save incomplete mappings and exit, but the replication engine cannot run until you have mapped all values. If you save incomplete mappings, the unmapped values are displayed with "(unmap)" in the Field Mappings pane, to indicate that you need to map them.

7. Click **OK** to save your entries and dismiss the dialog.
8. On the **Gateway Mappings** tab, click **Apply** to save your changes.

After you map select fields, the field pair is listed in the **Field Mappings** pane, indicated by "(map)". To view the value mappings, place the cursor over the field pair—the mappings are displayed in a tooltip. To change the values mapped for a field, click the field in the **Field Mappings** pane and click **Edit Value Mappings**. To remove a mapping, click the mapping in the **Field Mappings** pane, then click **Unmap**.

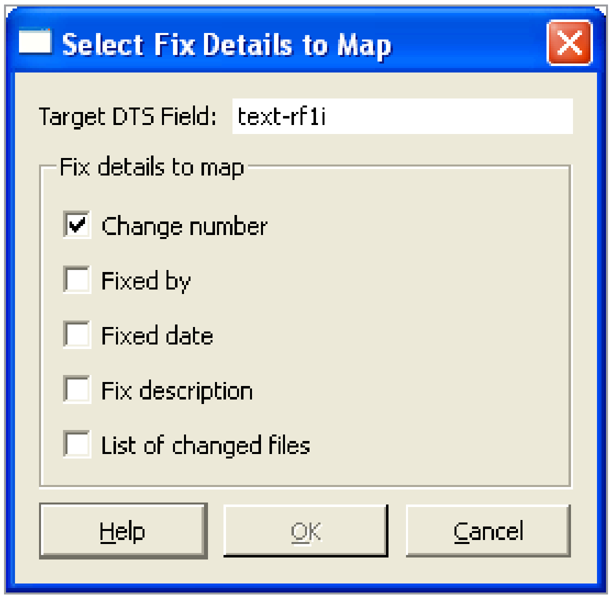
Mapping fix data

In Helix Server, you can associate a job with a changelist. When the changelist is submitted, the job status is changed, usually to a "closed" status, to indicate that the changes being submitted fix the problem reported in the associated job.

P4DTG enables you to configure how data from fixes submitted in Helix Server are transmitted to the defect tracker. To add fix mappings, click **Fix Details** in the **Perforce Fields** list.

To select the Helix Server data that is to be transmitted to the defect tracker when a job is fixed:

1. On the **Mappings** tab, scroll to the bottom of the **Field Mappings** list and select the desired mapping under **PERFORCE FIX DETAILS > COPIED > DEFECT TRACKER**. The Sel**ect Fix Details to Map** dialog is displayed, as shown in the following figure.


Select Fix Details to Map

Target DTS Field: text-rf1i

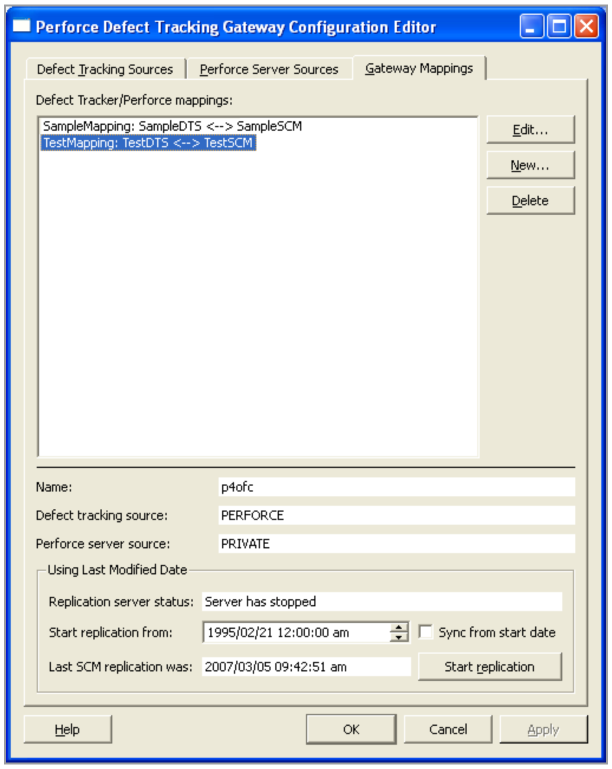
Fix details to map

- [✓] Change number
- [ ] Fixed by
- [ ] Fixed date
- [ ] Fix description
- [ ] List of changed files

2. Check the fields that you want transmitted to the defect tracker, then click **OK** to save your changes and dismiss the dialog.

Configuring and starting the replication engine

To configure and start the replication engine, display the **Gateway Mappings** tab, shown in the following figure.



Defect Tracker/Perforce mappings:

SampleMapping: SampleDTS <--> SampleSCM
TestMapping: TestDTS <--> TestSCM

Name: p4ofc

Defect tracking source: PERFORCE

Perforce server source: PRIVATE

Using Last Modified Date

Replication server status: Server has stopped

Start replication from: 1995/02/21 12:00:00 am ☐ Sync from start date

Last SCM replication was: 2007/03/05 09:42:51 am

For each mapping (Helix Server/defect tracker pair), you must run an instance of the replication engine, the program that propagates data between Helix Server and the defect tracker. Before you start the replication engine, you specify the starting date and time. The replication engine propagates new jobs and issues, and changes that possess a modification date equal to or greater than the specified time. The starting date and time defaults to the current system time. If you want to replicate older issues, set the starting date to the desired previous date and time.

When starting the replication for the first time, check **Sync from start date**.

To start the replication engine:

1. Specify the start date and time.
2. If you have stopped the replication engine and changed mappings, check **Sync from start date** to ensure that previously replicated jobs and issues are replicated again using your new mappings.
3. Click **Start replication**.

Checking for replication errors

The replication engine logs errors and warnings in the `p4dtg/repl` folder. For each instance of the replication engine, one log file is created. The log file is named `log-`***`mapfile`***`.log`, where ***`mapfile`*** is the name of the corresponding mapping.

After you stop an instance of the replication engine, be sure to examine the log file. If you are experiencing problems, check the log file. If you delete the log file, it is recreated the next time the replication engine is started.

If an error occurs during job-defect replication, the replication engine writes an error message to the **`DTG_ERROR`** field in the Helix Server job involved. Jobs containing entries in the **`DTG_ERROR`** field are not replicated. To re-enable replication, you must clear the **`DTG_ERROR`** field.

To list all jobs that have entries in the **`DTG_ERROR`** field, issue the following command:

```
p4 jobs -e "DTG_ERROR=*"
```

To detect replication problems on an ongoing basis, you can create a script that periodically issues the preceding command and reports any problems it detects.

If the replication engine encounters a problem that prevents it from successfully replicating data, it logs error information to a file named **`err-`*`mapfile`*** in the **`repl`** folder. The replication then exits and cannot be restarted until you resolve the problem and delete the error file.

Glossary

A

access level

A permission assigned to a user to control which commands the user can execute. See also the 'protections' entry in this glossary and the 'p4 protect' command in the P4 Command Reference.

admin access

An access level that gives the user permission to privileged commands, usually super privileges.

APC

The Alternative PHP Cache, a free, open, and robust framework for caching and optimizing PHP intermediate code.

archive

1. For replication, versioned files (as opposed to database metadata). 2. For the 'p4 archive' command, a special depot in which to copy the server data (versioned files and metadata).

atomic change transaction

Grouping operations affecting a number of files in a single transaction. If all operations in the transaction succeed, all the files are updated. If any operation in the transaction fails, none of the files are updated.

avatar

A visual representation of a Swarm user or group. Avatars are used in Swarm to show involvement in or ownership of projects, groups, changelists, reviews, comments, etc. See also the "Gravatar" entry in this glossary.

B

base

The file revision, in conjunction with the source revision, used to help determine what integration changes should be applied to the target revision.

binary file type

A Helix Server file type assigned to a non-text file. By default, the contents of each revision are stored in full, and file revision is stored in compressed format.

branch

(noun) A set of related files that exist at a specific location in the Perforce depot as a result of being copied to that location, as opposed to being added to that location. A group of related files is often referred to as a codeline. (verb) To create a codeline by copying another codeline with the 'p4 integrate', 'p4 copy', or 'p4 populate' command.

branch form

The form that appears when you use the 'p4 branch' command to create or modify a branch specification.

branch mapping

Specifies how a branch is to be created or integrated by defining the location, the files, and the exclusions of the original codeline and the target codeline. The branch mapping is used by the integration process to create and update branches.

branch view

A specification of the branching relationship between two codelines in the depot. Each branch view has a unique name and defines how files are mapped from the originating codeline to the target codeline. This is the same as branch mapping.

broker

Helix Broker, a server process that intercepts commands to the Helix Server and is able to run scripts on the commands before sending them to the Helix Server.

C

change review

The process of sending email to users who have registered their interest in changelists that include specified files in the depot.

changelist

A list of files, their version numbers, the changes made to the files, and a description of the changes made. A changelist is the basic unit of versioned work in Helix Server. The changes specified in the changelist are not stored in the depot until the changelist is submitted to the depot. See also atomic change transaction and changelist number.

changelist form

The form that appears when you modify a changelist using the 'p4 change' command.

changelist number

An integer that identifies a changelist. Submitted changelist numbers are ordinal (increasing), but not necessarily consecutive. For example, 103, 105, 108, 109. A pending changelist number might be assigned a different value upon submission.

check in

To submit a file to the Helix Server depot.

check out

To designate one or more files for edit.

checkpoint

A backup copy of the underlying metadata at a particular moment in time. A checkpoint can recreate db.user, db.protect, and other db.* files. See also metadata.

classic depot

A repository of Helix Server files that is not streams-based. The default depot name is depot. See also default depot and stream depot.

client form

The form you use to define a client workspace, such as with the 'p4 client' or 'p4 workspace' commands.

client name

A name that uniquely identifies the current client workspace. Client workspaces, labels, and branch specifications cannot share the same name.

client root

The topmost (root) directory of a client workspace. If two or more client workspaces are located on one machine, they should not share a client root directory.

client side

The right-hand side of a mapping within a client view, specifying where the corresponding depot files are located in the client workspace.

client workspace

Directories on your machine where you work on file revisions that are managed by Helix Server. By default, this name is set to the name of the machine on which your client workspace is located, but it can be overridden. Client workspaces, labels, and branch specifications cannot share the same name.

code review

A process in Helix Swarm by which other developers can see your code, provide feedback, and approve or reject your changes.

codeline

A set of files that evolve collectively. One codeline can be branched from another, allowing each set of files to evolve separately.

comment

Feedback provided in Helix Swarm on a changelist, review, job, or a file within a changelist or review.

commit server

A server that is part of an edge/commit system that processes submitted files (checkins), global workspaces, and promoted shelves.

conflict

1. A situation where two users open the same file for edit. One user submits the file, after which the other user cannot submit unless the file is resolved. 2. A resolve where the same line is changed when merging one file into another. This type of conflict occurs when the comparison of two files to a base yields different results, indicating that the files have been changed in different ways. In this case, the merge cannot be done automatically and must be resolved manually. See file conflict.

copy up

A Helix Server best practice to copy (and not merge) changes from less stable lines to more stable lines. See also merge.

counter

A numeric variable used to track variables such as changelists, checkpoints, and reviews.

CSRF

Cross-Site Request Forgery, a form of web-based attack that exploits the trust that a site has in a user's web browser.

D

default changelist

The changelist used by a file add, edit, or delete, unless a numbered changelist is specified. A default pending changelist is created automatically when a file is opened for edit.

deleted file

In Helix Server, a file with its head revision marked as deleted. Older revisions of the file are still available. in Helix Server, a deleted file is simply another revision of the file.

delta

The differences between two files.

depot

A file repository hosted on the server. A depot is the top-level unit of storage for versioned files (depot files or source files) within a Helix Core Server. It contains all versions of all files ever submitted to the depot. There can be multiple depots on a single installation.

depot root

The topmost (root) directory for a depot.

depot side

The left side of any client view mapping, specifying the location of files in a depot.

depot syntax

Helix Server syntax for specifying the location of files in the depot. Depot syntax begins with: //depot/

diff

(noun) A set of lines that do not match when two files are compared. A conflict is a pair of unequal diffs between each of two files and a base. (verb) To compare the contents of files or file revisions. See also conflict.

donor file

The file from which changes are taken when propagating changes from one file to another.

E

edge server

A replica server that is part of an edge/commit system that is able to process most read/write commands, including 'p4 integrate', and also deliver versioned files (depot files).

exclusionary access

A permission that denies access to the specified files.

exclusionary mapping

A view mapping that excludes specific files or directories.

F

file conflict

In a three-way file merge, a situation in which two revisions of a file differ from each other and from their base file. Also, an attempt to submit a file that is not an edit of the head revision of the file in the depot, which typically occurs when another user opens the file for edit after you have opened the file for edit.

file pattern

Helix Server command line syntax that enables you to specify files using wildcards.

file repository

The master copy of all files, which is shared by all users. In Helix Server, this is called the depot.

file revision

A specific version of a file within the depot. Each revision is assigned a number, in sequence. Any revision can be accessed in the depot by its revision number, preceded by a pound sign (#), for example testfile#3.

file tree

All the subdirectories and files under a given root directory.

file type

An attribute that determines how Helix Server stores and diffs a particular file. Examples of file types are text and binary.

fix

A job that has been closed in a changelist.

form

A screen displayed by certain Helix Server commands. For example, you use the change form to enter comments about a particular changelist to verify the affected files.

forwarding replica

A replica server that can process read-only commands and deliver versioned files (depot files). One or more replicate servers can significantly improve performance by offloading some of the master server load. In many cases, a forwarding replica can become a disaster recovery server.

G

Git Fusion

A Perforce product that integrates Git with Helix, offering enterprise-ready Git repository management, and workflows that allow Git and Helix Server users to collaborate on the same projects using their preferred tools.

graph depot

A depot of type graph that is used to store Git repos in the Helix Server. See also Helix4Git.

Gravatar

gravatar.com is a third party service that you can subscribe to, gravatar enables you to upload an image that you can use in Swarm. When configured, Swarm will attempt to fetch your avatar from gravatar.com and use it within Swarm. If your avatar is not found on gravatar.com, Swarm will use one of its own default avatars to represent your activity. See also the "avatar" entry in this glossary.

group

A feature in Helix Server that makes it easier to manage permissions for multiple users.

H

have list

The list of file revisions currently in the client workspace.

head revision

The most recent revision of a file within the depot. Because file revisions are numbered sequentially, this revision is the highest-numbered revision of that file.

Helix Server

The Helix Server depot and metadata; also, the program that manages the depot and metadata, also called Helix Core Server.

Helix TeamHub

A Perforce management platform for code and artifact repository. TeamHub offers built-in support for Git, SVN, Mercurial, Maven, and more.

Helix4Git

Perforce solution for teams using Git. Helix4Git offers both speed and scalability and supports hybrid environments consisting of Git repositories and 'classic' Helix Server depots.

I

iconv

iconv is a PHP extension that performs character set conversion, and is an interface to the GNU libiconv library.

integrate

To compare two sets of files (for example, two codeline branches) and determine which changes in one set apply to the other, determine if the changes have already been propagated, and propagate any outstanding changes from one set to another.

J

job

A user-defined unit of work tracked by Helix Server. The job template determines what information is tracked. The template can be modified by the Helix Server system administrator. A job describes work to be done, such as a bug fix. Associating a job with a changelist records which changes fixed the bug.

job daemon

A job daemon is a program that checks the Helix Server machine daily to determine if any jobs are open. If so, the daemon sends an email message to interested users, informing them the number of jobs in each category, the severity of each job, and more.

job specification

A form describing the fields and possible values for each job stored in the Helix Server machine.

job view

A syntax used for searching Helix Server jobs.

journal

A file containing a record of every change made to the Helix Server's metadata since the time of the last checkpoint. This file grows as each Helix Server transaction is logged. The file should be automatically truncated and renamed into a numbered journal when a checkpoint is taken.

journal rotation

The process of renaming the current journal to a numbered journal file.

journaling

The process of recording changes made to the Helix Server's metadata.

L

label

A named list of user-specified file revisions.

label view

The view that specifies which filenames in the depot can be stored in a particular label.

lazy copy

A method used by Helix Server to make internal copies of files without duplicating file content in the depot. A lazy copy points to the original versioned file (depot file). Lazy copies minimize the consumption of disk space by storing references to the original file instead of copies of the file.

license file

A file that ensures that the number of Helix Server users on your site does not exceed the number for which you have paid.

list access

A protection level that enables you to run reporting commands but prevents access to the contents of files.

local depot

Any depot located on the currently specified Helix Server.

local syntax

The syntax for specifying a filename that is specific to an operating system.

lock

1. A file lock that prevents other clients from submitting the locked file. Files are unlocked with the 'p4 unlock' command or by submitting the changelist that contains the locked file. 2. A database lock that prevents another process from modifying the database db.* file.

log

Error output from the Helix Server. To specify a log file, set the P4LOG environment variable or use the p4d -L flag when starting the service.

M

mapping

A single line in a view, consisting of a left side and a right side that specify the correspondences between files in the depot and files in a client, label, or branch. See also workspace view, branch view, and label view.

MDS checksum

The method used by Helix Server to verify the integrity of versioned files (depot files).

merge

1. To create new files from existing files, preserving their ancestry (branching). 2. To propagate changes from one set of files to another. 3. The process of combining the contents of two conflicting file revisions into a single file, typically using a merge tool like P4Merge.

merge file

A file generated by the Helix Server from two conflicting file revisions.

metadata

The data stored by the Helix Server that describes the files in the depot, the current state of client workspaces, protections, users, labels, and branches. Metadata includes all the data stored in the Perforce service except for the actual contents of the files.

modification time or modtime

The time a file was last changed.

MPM

Multi-Processing Module, a component of the Apache web server that is responsible for binding to network ports, accepting requests, and dispatch operations to handle the request.

N

nonexistent revision

A completely empty revision of any file. Syncing to a nonexistent revision of a file removes it from your workspace. An empty file revision created by deleting a file and the #none revision specifier are examples of nonexistent file revisions.

numbered changelist

A pending changelist to which Helix Server has assigned a number.

O

opened file

A file that you are changing in your client workspace that is checked out. If the file is not checked out, opening it in the file system does not mean anything to the versioning engineer.

owner

The Helix Server user who created a particular client, branch, or label.

P

p4

1. The Helix Core Server command line program. 2. The command you issue to execute commands from the operating system command line.

p4d

The program that runs the Helix Server; p4d manages depot files and metadata.

P4PHP

The PHP interface to the Helix API, which enables you to write PHP code that interacts with a Helix Server machine.

PECL

PHP Extension Community Library, a library of extensions that can be added to PHP to improve and extend its functionality.

pending changelist

A changelist that has not been submitted.

project

In Helix Swarm, a group of Helix Server users who are working together on a specific codebase, defined by one or more branches of code, along with options for a job filter, automated test integration, and automated deployment.

protections

The permissions stored in the Helix Server’s protections table.

proxy server

A Helix Server that stores versioned files. A proxy server does not perform any commands. It serves versioned files to Helix Server clients.

R

RCS format

Revision Control System format. Used for storing revisions of text files in versioned files (depot files). RCS format uses reverse delta encoding for file storage. Helix Server uses RCS format to store text files. See also reverse delta storage.

read access

A protection level that enables you to read the contents of files managed by Helix Server but not make any changes.

remote depot

A depot located on another Helix Server accessed by the current Helix Server.

replica

A Helix Server that contains a full or partial copy of metadata from a master Helix Server. Replica servers are typically updated every second to stay synchronized with the master server.

repo

A graph depot contains one or more repos, and each repo contains files from Git users.

reresolve

The process of resolving a file after the file is resolved and before it is submitted.

resolve

The process you use to manage the differences between two revisions of a file. You can choose to resolve conflicts by selecting the source or target file to be submitted, by merging the contents of conflicting files, or by making additional changes.

reverse delta storage

The method that Helix Server uses to store revisions of text files. Helix Server stores the changes between each revision and its previous revision, plus the full text of the head revision.

revert

To discard the changes you have made to a file in the client workspace before a submit.

review access

A special protections level that includes read and list accesses and grants permission to run the p4 review command.

review daemon

A review daemon is a program that periodically checks the Helix Server machine to determine if any changelists have been submitted. If so, the daemon sends an email message to users who have subscribed to any of the files included in those changelists, informing them of changes in files they are interested in.

revision number

A number indicating which revision of the file is being referred to, typically designated with a pound sign (#).

revision range

A range of revision numbers for a specified file, specified as the low and high end of the range. For example, myfile#5,7 specifies revisions 5 through 7 of myfile.

revision specification

A suffix to a filename that specifies a particular revision of that file. Revision specifiers can be revision numbers, a revision range, change numbers, label names, date/time specifications, or client names.

RPM

RPM Package Manager is a tool, and package format, for managing the installation, updates, and removal of software packages for Linux distributions such as Red Hat Enterprise Linux, the Fedora Project, and the CentOS Project.

S

server data

The combination of server metadata (the Helix Server database) and the depot files (your organization's versioned source code and binary assets).

server root

The topmost directory in which p4d stores its metadata (db.* files) and all versioned files (depot files or source files). To specify the server root, set the P4ROOT environment variable or use the p4d -r flag.

service

In the Helix Core Server, the shared versioning service that responds to requests from Helix Server client applications. The Helix Server (p4d) maintains depot files and metadata describing the files and also tracks the state of client workspaces.

shelve

The process of temporarily storing files in the Helix Server without checking in a changelist.

status

For a changelist, a value that indicates whether the changelist is new, pending, or submitted. For a job, a value that indicates whether the job is open, closed, or suspended. You can customize job statuses. For the 'p4 status' command, by default the files opened and the files that need to be reconciled.

stream

A branch with additional intelligence that determines what changes should be propagated and in what order they should be propagated.

stream depot

A depot used with streams and stream clients.

submit

To send a pending changelist into the Helix Server depot for processing.

super access

An access level that gives the user permission to run every Helix Server command, including commands that set protections, install triggers, or shut down the service for maintenance.

symlink file type

A Helix Server file type assigned to symbolic links. On platforms that do not support symbolic links, symlink files appear as small text files.

sync

To copy a file revision (or set of file revisions) from the Helix Server depot to a client workspace.

T

target file

The file that receives the changes from the donor file when you integrate changes between two codelines.

text file type

Helix Server file type assigned to a file that contains only ASCII text, including Unicode text. See also binary file type.

theirs

The revision in the depot with which the client file (your file) is merged when you resolve a file conflict. When you are working with branched files, theirs is the donor file.

three-way merge

The process of combining three file revisions. During a three-way merge, you can identify where conflicting changes have occurred and specify how you want to resolve the conflicts.

trigger

A script automatically invoked by Helix Server when various conditions are met. (See "Helix Versioning Engine Administrator Guide: Fundamentals" on "Using triggers to customize behavior")

two-way merge

The process of combining two file revisions. In a two-way merge, you can see differences between the files.

typemap

A table in Helix Server in which you assign file types to files.

U

user

The identifier that Helix Server uses to determine who is performing an operation.

V

versioned file

Source files stored in the Helix Server depot, including one or more revisions. Also known as a depot file or source file. Versioned files typically use the naming convention 'filenamev' or '1.changelist.gz'.

view

A description of the relationship between two sets of files. See workspace view, label view, branch view.

W

wildcard

A special character used to match other characters in strings. The following wildcards are available in Helix Server: * matches anything except a slash; ... matches anything including slashes; %%0 through %%9 is used for parameter substitution in views.

workspace

See client workspace.

workspace view

A set of mappings that specifies the correspondence between file locations in the depot and the client workspace.

write access

A protection level that enables you to run commands that alter the contents of files in the depot. Write access includes read and list accesses.

X

XSS

Cross-Site Scripting, a form of web-based attack that injects malicious code into a user's web browser.

Y

yours

The edited version of a file in your client workspace when you resolve a file. Also, the target file when you integrate a branched file.

License statements