Helix Web Services Guide

2016.1.0 Alpha

Tristan Juricek **tjuricek@perforce.com**

Table of Contents

About This Manual

This guide covers administering Helix Web Services, and, provides a reference for developing applications using it. The guide is largely a reference of APIs and infrequent tasks, such as installation or configuration. It is not the only guide required for development or installation. Helix Web Services extends other services like the Helix Versioning Engine and Git Fusion. This guide will not cover many details specific to other services.

Please give us feedback

If you have any feedback for us, or detect any errors in this guide, please email details to manual@perforce.com.

Overview

Helix Web Services, or HWS, is a middleware platform for Perforce technologies in the Helix ecosystem. HWS forms a central point for applications to communicate and coordinate across service applications installed in a single cluster. While HWS does contain significant logic, it is an extension to other technologies, and is unlikely to be used in isolation.

Knowledge Required

For installing or configuring an HWS instance, you should be familiar with common system administration tasks and conventions. To develop applications against HWS, we assume you are familiar with the HTTP protocol. We do provide some basic client API libraries, but these are to assist in making it easy to get started, not to abstract away or hide the fundamentals of the HTTP-based system.

To secure your HWS instances, you will need to be comfortable with TLS certificates. Helix Web Services recommends using HTTPS for connections, unless you can secure access to the server by other means. For evaluation, we do simplify the process of using a self-signed certificate. However, for production, it is recommended you purchase and deploy a properly signed certificate.

Additionally, because this guide interfaces with the Helix Versioning Engine, we assume you are familiar with it. For more information about the Helix Versioning Engine, consult the following online guides:

- Product overview: http://www.perforce.com/versioning-engine

- P4 Command Reference: http://www.perforce.com/perforce/doc.current/manuals/cmdref/index.html

- Admin fundamentals: http://www.perforce.com/perforce/doc.current/manuals/p4sag/index.html

Release compatibility of the API

The major and minor version of Helix Web Services is associated with a major release of the Helix Versioning Engine (p4d). For example, the version 2016.1.0 is associated with the 2016.1 release of p4d, and what we recommend to deploy. The APIs that you develop against will use data documented as if this was your runtime system.

The Helix Web Services APIs are split into default, or stable, methods and unstable alpha methods. Methods that are stable should work against future releases of Helix Web Services, for (at least) the next 3 major releases. A major release is when we update the major or minor version, e.g., 2016.2.0 is the next major release after 2016.1.0. That is, a client application written against 2016.1.0 can use that SDK against Helix Web Services releases 2016.2.0, 2017.1.0, and 2017.2.0. Unstable methods may stop working with an upgrade, so planning and testing is required if you put these methods into use. If you use our SDKs, the use of alpha methods is quite explicit, so generally easy to track.

You can use Helix Web Services against older versions of the Helix Versioning Engine, even though the API was developed against a newer release. In these cases, the entire API is not guaranteed to exist, and you will need to test your applications against these server configurations. Some fields may be missing, and some behaviors may change subtly. In general, however, we try to assist in making these scenarios possible, and do not explicitly break with older versions of p4d. It's recommended to set the `P4APILEVEL` setting for each server, or globally to the oldest version in use. See for more details on server configuration.

Deploying Helix Web Services

Obtaining HWS

> **Warning**
>
> We have not yet defined exactly how people will download HWS. It is likely that we'll have a section under plugins & integrations, or server utilities. Once this is defined, we'll need to complete this section.

Quick Start of HWS

Before you can install Helix Web Services, you will need to ensure Java 8 has been installed and configured on your machine. If you intend to connect to p4d over TLS, you will need to additionally configure the Java Cryptography Extensions.

> **Warning**
>
> Java 8 installation varies from OS to OS. We probably should guide users to good instructions; Oracle does a bad job of this.

Depending upon your operating system, you will have a slightly different experience getting started with Helix Web Services.

-

-

-

Quick Start installing via Linux Packages

Warning	We do not currently host the linux packages via a package manager.

1. Follow the instructions on https://www.perforce.com/perforce-packages to configure your APT or YUM repository manager.

2. Install the `helix-web-services` package

   a. On APT-based distributions, this is `sudo apt-get install helix-web-services`

   b. On YUM-based distributions, this is `sudo yum install helix-web-services`

3. Execute the configuration script: `sudo /opt/perforce/helix-web-services/sbin/configure-helix-web-services`

4. Optionally, edit the configuration file `./etc/helix-web-services.conf` with any customizations you want.

Tip	You probably want to set the `P4PORT` and `P4CHARSET` variables, see "Configuration" on page 23.

5. Start the server

   a. On Linux, this is typically `sudo service helix_web_services start`

   b. On OS X, you'll probably need to run `sudo launchctl load -w /Library/LaunchDaemons/com.perforce.helix_web_services.server.WebApp.plist`

Quick Start of HWS on Linux or OS X using the Binary Tarball Distribution

Provided you have the binary tarball distribution, `helix-web-services-bin.tar.gz`, you can set it up on your system quickly using the following steps:

1. Expand the tarball, typically in a place like `/opt/perforce`:

   ```
   $ sudo mkdir -p /opt/perforce
   $ cd /opt/perforce
   $ sudo tar xzf /path/to/helix-web-services-bin.tgz
   ```

2. Run our configuration script:

   ```
   $ sudo ./sbin/configure-helix-web-services
   ```

3. Optionally, edit the configuration file `./etc/helix-web-services.conf` with any customizations you want.

Tip	You probably want to set the `P4PORT` and `P4CHARSET` variables, see "Configuration" on page 23.

4. Start the server

a. On Linux, this is typically `sudo service helix_web_services start`

b. On OS X, you'll probably need to run `sudo launchctl load -w /Library/LaunchDaemons/ com.perforce.helix_web_services.server.WebApp.plist`

Quick Start of HWS on Windows using the Zipfile Distribution

Provided you have obtained a copy of `helix-web-services.zip`, here's how you can start up an instance quickly:

1. Extract the archives of the .zip to a directory, e.g., `C:\helix-web-services`

2. Open a Command Prompt as Administrator

3. In this command prompt, run the configuration script:

   ```
   cd C:\helix-web-services
   sbin\configure-helix-web-services.exe
   ```

4. Optionally, edit the configuration file `C:\helix-web-services\etc\helix-web-services.conf` with any customizations you want.

Tip	You probably want to set the `P4PORT` and `P4CHARSET` variables, see "Configuration" on page 23.

5. Open your service control manager, and double check that "Helix Web Services" is listed and running.

Uninstalling Helix Web Services

To completely uninstall Helix Web Services from your system, a simple console script has been provided.

On Unix machines this is typically called with the command:

```
$ sudo /opt/perforce/helix-web-services/sbin/uninstall-helix-web-services
```

(Substitute the directory `/opt/perforce/helix-web-services` for your installation directory if different.)

On Windows, you will run the `sbin/uninstall-helix-web-services.exe` application with Administrator privileges.

This will completely remove the installation directory from your system. If you have configuration you wish to retain, please make a copy of that configuration outside of the installation location.

Configuration

Configuration of Helix Web Services is split into a main file, and individual files that indicate each p4d individually, see "P4D Configuration" on page 26. You will need both the main file and at least one p4d configuration.

Main Helix Web Services Settings

Most of these settings are available in the configuration file typically installed within the package at
/opt/perforce/helix-web-services/etc/helix-web-services.conf, or in the subdirectory etc of
wherever you unpacked the binary archive distribution.

Variable	Type	Description	Default
ACCESS_CONTROL_ALLOW_O	String	Value for the **Access-Control-Allow-Origin** CORS header.	*
ACCESS_CONTROL_ALLOW_H	String	Value for the **Access-Control-Allow-Headers** CORS header.	*
ACCESS_CONTROL_REQUEST	String	Value for the **Access-Control-Request-Method** CORS header.	*
AUTO_TRUST	Boolean	Enable auto-trusting all new servers that this service is connecting to. If previously trusted will throw an error if fingerprint changed	false
COMMAND_WHITELIST	Array	Allows access to run commands via "GET /api/2016.1.0/{server}/commands/{command}" on page 473 or "POST /api/2016.1.0/{server}/commands/{command}" on page 474 Each element in the array is either the name of a command to allow, or an array of the command name and any required arguments.	["info", ["files", "-m"]]
DEFAULT_API_LEVEL	String	The api level (e.g., "80") to use for P4 connections. If set, we'll use this instead of associating it with the	

Variable	Type	Description	Default
		platform version we're targeting.	
`ENABLE_GIT_FUSION`	Boolean	Switch to enable GitFusion endpoints.	false
`ENABLE_HTTPS`	Boolean	Configure the web server to use HTTPS. You must configure the keystore file: see `KEYSTORE_FILE` setting. Related optional settings are: `KEYSTORE_PASSWORD`, `TRUSTSTORE_FILE`, and `TRUSTSTORE_PASSWORD`.	false
`GITFUSIONDEPOT`	String	The depot name for Git Fusion data. This makes no sense to use if `ENABLE_GIT_FUSION` is false.	.git-fusion
`HWS_PORT`	Number	The port to respond to for new connections.	9000
`KEYSTORE_FILE`	String	The keystore file for secure connections.	
`KEYSTORE_PASSWORD`	String	The password for the keystore.	
`MAX_SERVER_CONNECTIONS`	Number	The maximum number of connections we'll allow this instance to acquire per P4PORT	50
`P4DCONFIGDIR`	String	The local directory where p4d configuration settings files are stored. These are yaml files, named with the the value of `P4DID`.	
`TRUST_FINGERPRINTS`	String	Local file path that indicates a list of allowed SSL fingerprints.	

Variable	Type	Description	Default
TRUSTSTORE_FILE	String	The path to the truststore file, if empty, will reuse the keystore.	
TRUSTSTORE_PASSWORD	String	The password for the truststore file.	

P4D Configuration

Each instance of p4d you use should be registered in by creating a file in the P4DCONFIGDIR, set in the main configuration file.

Each p4d configuration file is a YAML document with the following properties:

Property	Required	Description
id	Yes	The ID of the settings, which should be the name of the file and the value you use for P4DID to find these settings. Please do not use complex characters for this ID, simple ASCII alphanumeric strings are recommended.
name	No	A string suitable for display to the user
description	No	A free text field to indicate the connection's purpose.
P4PORT	Yes	The host and ID, optionally preceded by ssl: if it is an TLS enabled server.
P4CHARSET	No	The character set of the server connections to use, e.g., utf8
APILEVEL	No	Specify the API level to connect to this server as, e.g., 80, for 2016.1 p4d.

For example, here is a configuration for the default ID that should be in a file called default. In the default installation this is /opt/perforce/helix-web-services/etc/p4d/default.

default.

```
id: default
name: Default
description: Our typical default server
P4PORT: 127.0.0.1:1666
P4CHARSET: utf8
```

Client Application Development

Authentication

Almost every Helix Web Services method requires authentication. We provide a simple token-based authentication scheme, similar to what you might use with OAuth. You must authenticate using our "POST /api/2016.1.0/login" on page 465 method, then use that ticket value as the `Authorization` header value.

Error Conventions

Upon success, methods of Helix Web Services return the 200 HTTP status code. If your authentication token is not valid, you will receive the status code 403. You will then need to update the login token (using "POST /api/2016.1.0/login" on page 465).

HTTP status code 401 will likely originate from the Helix Versioning Engine. In this case, we provide information to the client application via a JSON object. This object will contain three main properties:

Property	Description
`MessageCode`	A numeric ID for the problem, typically defined by the Helix Versioning Engine. Should be the same value as Error::GetGeneric().
`MessageSeverity`	Mirrors the severity levels of Error::GetSeverity(). Generally going to be 3 or 4.
`MessageText`	Informational text that will likely not be localized appropriately for the user.

When 500 errors happen, a serious problem has occurred, which may mean an important subsystem is compromised. Do not expect a response body. The problem will have to be investigated on the server side. Your client should not attempt further communication with Helix Web Services.

Client SDK Reference Guides

Java SDK Reference

The Java SDK uses the `ApiClient` to initialize and configure access to the server. From the `ApiClient` object, you obtain interfaces to access the individual methods. The two interfaces are `DefaultApi`,

which contain stable methods, and the `AlphaApi`, which are unstable methods (i.e., likely to change in future releases).

The Java SDK requires Java 8.

Getting Started

In the `clients/java/lib` directory is our Jar and all of it's dependencies. Include these jars in your application's classpath to start running.

Typically, you will use the `ApiClient` to sign in and generate a handle to the `DefaultApi`, which you use to interact with the HWS server:

```
import com.perforce.helix_web_services_client.*;
import com.perforce.helix_web_services_client.models.*;

// Note: by default we create a self-signed certificate. If you run into SSL validation
// errors, it's likely that you'll need to create a "test mode" API client using
// the constructor ApiClient(boolean, String), e.g.,
//   new ApiClient(true, "https://mycompany.example.com");
// You shouldn't use that constructor in production applications. Either ditch SSL or
// setup your trust store.
ApiClient apiClient = new ApiClient("https://mycompany.example.com");

DefaultApi api = apiClient.createDefaultService();

// Create an authentication ticket to use as our security token.
//
// This ticket does create p4 tickets internally, we do not expose those tickets.
LoginRequest loginRequest = new LoginRequest();
loginRequest.setUser("jdoe");
loginRequest.setPassword("johndoe1A!");

LoginResponse loginResponse = defaultApi.loginPost(loginRequest);

apiClient.setApiKey(loginResponse.getTicket());

// Now, perform operations on behalf of your user.
//
// This is a typical "command-style" method that returns results similar to
// the output of "p4 depots".
List<DepotsCommand> depots = api.serverDepotsGet("myserver");

// Methods that operate on specifications typically return "Spec" model types,
// in this case, based on the `p4 user -o jdoe` command.
UserCommand user = api.serverUsersUserGet(p4dId, "jdoe");
```

ApiClient Reference

ApiClient.createWithTicket

Create an ApiClient and sign into a Helix Web Services server.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
ApiClient ApiClient.createWithTicket(String baseUrl, String user, String password)
```

Table 1. Parameters

Type	Name	Description
String	baseUrl	The Helix Web Services base URL, e.g., "https:// mycompany.example.com"
String	user	The user login. If you have a multiple p4d configuration in your web services, this login is used against all servers.
String	password	The user's password. If you have a multiple p4d configuration, this password is used against all servers.

Java Example.

```
ApiClient apiClient = ApiClient.createWithTicket("https://myserver.example.com", "jdoe",
  "jdoePassword");
```

ApiClient Constructors

ApiClient#ApiClient(String basePath)

Construct an ApiClient with the indicated basePath.

If you are testing out a new installation, this will likely not work due to the use of a self-signed certificate on the server.

> **Warning**
> This ApiClient is **not** authenticated after construction.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
ApiClient(String basePath)
```

Table 2. Parameters

Type	Name	Description
`String`	`basePath`	The base URL to Helix Web Services instance, e.g., https://myserver.example.com.

Java Example.

```
ApiClient apiClient = new ApiClient("https://myserver.example.com");
```

ApiClient#ApiClient(boolean trustAllSsl, String basePath)

A special construction option that allows your client to trust all certificates.

By default, all HWS instances start with a self-signed certificate. You may need to construct using this variation to validate your code is working in a non-production system.

> **Warning**
>
> This ApiClient is **not** authenticated after construction.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
ApiClient(boolean trustAllSsl, String basePath)
```

Table 3. Parameters

Type	Name	Description
`boolean`	`trustAllSsl`	When `true`, we'll disable all SSL/TLS certificate verification. This may be needed for testing with self-signed certificates, though we recommend avoiding this in production.
`String`	`basePath`	The base URL to Helix Web Services instance, e.g., https://myserver.example.com.

Java Example.

```
// Create a very insecure ApiClient.
ApiClient apiClient = new ApiClient(true, "https://myserver.example.com");
```

ApiClient#createDefaultService

Obtain a handle to the service interface for stable methods.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
DefaultApi ApiClient#createDefaultService(boolean trustAllSsl, String basePath)
```

Table 4. Returns

Type	Notes
"DefaultApi Reference" on page 34	The main interface your application should use to interact with Helix Web Services.

ApiClient#createService

Obtain a handle to a service interface. This can specify any particular API, such as the "AlphaApi Reference" on page 71.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
API ApiClient#createService(Class<API> interfaceName)
```

Table 5. Parameters

Type	Name	Description
Class<API>	serviceClass	The interface class to use.

Table 6. Returns

Type	Notes
The object handle that is the type specified by the parameter.	The main interface your application should use to interact with Helix Web Services.

Java Example.

```
AlphaApi alphaApi = apiClient.createService(AlphaApi.class);
```

ApiClient#getBasePath

Returns the base URL to your HWS instance.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
String ApiClient#getBasePath()
```

Table 7. Returns

Type	Notes
String	The base URL that's generally been specified by the constructor.

Java Example.

```
String url = apiClient.getBasePath();
```

ApiClient#getStatus

Returns the current status of the system. Unlike the "ApiClient#isOK" on page 32 method, this **will** throw an exception in the face of any kind of network failure.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
HWSStatus ApiClient#getStatus()
```

Table 8. Returns

Type	Notes
"HWSStatus" on page 101	

ApiClient#isOK

Checks the status and returns true if Helix Web Services is responding and doesn't report any problems.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
boolean ApiClient#isOK()
```

Table 9. Returns

Type	Notes
boolean	True if HWS responds, false otherwise. We generally will **not** throw exceptions in the face of failures.

Java Example.

```
if (!apiClient.isOK()) {
    throw new IllegalStateException("Helix Web Services is not available");
}
```

ApiClient#isSupported

Will fetch the supported list of versions from the Helix Web Services server, and verify if this Client SDK version is supported by this server. In general, it's probably a good idea to use this method when configuring a new Helix Web Services server, in order to say, let the user know if they need to upgrade their application.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
boolean ApiClient#isSupported()
```

Table 10. Returns

Type	Notes
boolean	True if the client is supported, false if it's not or there's any kind of problem connecting to the server.

Java Example.

```
if (!apiClient.isSupported()) {
    throw new IllegalStateException("Helix Web Services doees not support this client version or
 is not available");
}
```

ApiClient#setApiKey

Sets the per-user authentication ticket to use for access to Helix Web Services.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
void ApiClient#setApiKey(String apiKey)
```

Table 11. Parameters

Type	Name	Description
String	apiKey	The ticket value of the "LoginResponse" on page 105.

DefaultApi Reference

DefaultApi#configP4dsGet

The list of registered p4d servers in your cluster.

This is provided by a special set of configuration files in the system. For more information, consult the Helix Web Services user guide.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<P4dConfigId> DefaultApi#configP4dsGet()
```

Table 12. Returns

Type	Notes
List< "P4dConfigId" on page 105 >	

DefaultApi#loginPost

Logs into the primary authentication source.

This can either be a p4d instance or Helix Cloud, depending upon the configuration of your Helix Web Services instance.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
LoginResponse DefaultApi#loginPost(LoginRequest loginRequest)
```

Table 13. Parameters

Type	Name	Description
"LoginRequest" on page	loginRequest	The user login and password.

Table 14. Returns

Type	Notes
"LoginResponse" on page 105	Object with ticket to use for Basic auth password.

DefaultApi#statusGet

A simple structure to monitor for "problems" an admin should take care of, and, report the current application version.

This method does not require authentication.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
HWSStatus DefaultApi#statusGet()
```

Table 15. Returns

Type	Notes
"HWSStatus" on page 101	

DefaultApi#serverBranchesGet

Lists available branches in the system. The resources of this list are summaries of branches in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<BranchesCommand> DefaultApi#serverBranchesGet(String server)
```

Table 16. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 17. Returns

Type	Notes
List< "BranchesCommand" on page 76 >	Summaries of branches in the system.

DefaultApi#serverBranchesPost

Creates a new branch specification, like the `p4 branch` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverBranchesPost(String server, BranchCommand body)
```

Table 18. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
"BranchCommand" on p	body	The branch specification.

Table 19. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverBranchesBranchGet

Returns the branch spec details of the particular branch.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
BranchCommand DefaultApi#serverBranchesBranchGet(String server, String branch)
```

Table 20. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	branch	The branch ID

Table 21. Returns

Type	Notes
"BranchCommand" on page 75	Branch spec details

DefaultApi#serverBranchesBranchPatch

Update branch specifications, similar to the `p4 branch` command. Only the specified parameters in the body will be changed.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverBranchesBranchPatch(String server, String branch, BranchCommand
  body)
```

Table 22. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	branch	The branch ID
"BranchCommand" on p	body	Fields of the branch to update

Table 23. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverBranchesBranchDelete

Removes the branch specification, similar to the `p4 branch -d` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverBranchesBranchDelete(String server, String branch)
```

Table 24. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	branch	The branch ID

Table 25. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverChangesGet

Lists available changes in the system. The resources of this list are summaries of changes in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<ChangesCommand> DefaultApi#serverChangesGet(String server, Integer max, String status,
 String user, String files)
```

Table 26. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
Integer	max	Limit the number of change results

Type	Name	Description
String	status	The status of the changes, e.g., `submitted`
String	user	The user's login who submitted the change
String	files	Limit changes to the depot path expressions. See the changes command description.

Table 27. Returns

Type	Notes
List< "ChangesCommand" on page 79 >	Summaries of changes in the system.

DefaultApi#serverChangesChangeGet

Returns the change spec details of the particular change.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
ChangeCommand DefaultApi#serverChangesChangeGet(String server, String change)
```

Table 28. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	change	The change ID

Table 29. Returns

Type	Notes
"ChangeCommand" on page 76	Change spec details

DefaultApi#serverClientsGet

Lists available clients in the system. The resources of this list are summaries of clients in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<ClientsCommand> DefaultApi#serverClientsGet(String server)
```

Table 30. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 31. Returns

Type	Notes
List< ["ClientsCommand" on page 84](#) >	Summaries of clients in the system.

DefaultApi#serverClientsPost

Creates a new client specification, like the `p4 client` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverClientsPost(String server, ClientCommand client)
```

Table 32. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
["ClientCommand" on pa](#)	client	The client spec

Table 33. Returns

Type	Notes
["CommandResponse" on page 87](#)	

DefaultApi#serverClientsClientGet

Returns the client spec details of the particular client.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
ClientCommand DefaultApi#serverClientsClientGet(String server, String client)
```

Table 34. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	client	The client ID

Table 35. Returns

Type	Notes
"ClientCommand" on page 80	Client spec details

DefaultApi#serverClientsClientPatch

Update client specifications, similar to the `p4 client` command. Only the specified parameters in the body will be changed.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverClientsClientPatch(String server, String client, ClientCommand
  body)
```

Table 36. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	client	The client ID

Type	Name	Description
"ClientCommand" on pa	body	Fields of the client to update

Table 37. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverClientsClientDelete

Removes the client specification, similar to the `p4 client -d` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverClientsClientDelete(String server, String client)
```

Table 38. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	client	The client ID

Table 39. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverCommandsCommandGet

Execute a Perforce command that requires no input. This only allows commands that have been whitelisted on your system. See the "Configuration" on page 23 section for details.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverCommandsCommandGet(String server, String command,  arg)
```

Table 40. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	command	The command name
	arg	Command arguments

Table 41. Returns

Type	Notes
"CommandResponse" on page 87	Generic list of hashes response

DefaultApi#serverCommandsCommandPost

Execute a Perforce command that accepts input, like a spec. This only allows commands that have been whitelisted on your system. See the "Configuration" on page 23 section for details.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverCommandsCommandPost(String server, String command,  arg,
 CommandRequest input)
```

Table 42. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	command	The command name
	arg	Command arguments
"CommandRequest" on	input	A hash used as input to the command

Table 43. Returns

Type	Notes
"CommandResponse" on page 87	Generic list of hashes response

DefaultApi#serverCountersGet

Lists available counters in the system. The resources of this list are summaries of counters in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<Counter> DefaultApi#serverCountersGet(String server)
```

Table 44. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 45. Returns

Type	Notes
List< [“Counter” on page 88](#) >	Summaries of counters in the system.

DefaultApi#serverCountersCounterPut

Update counter specifications, similar to the `p4 counter` command. Only the specified parameters in the body will be changed.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverCountersCounterPut(String server, String counter, Counter body)
```

Table 46. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	counter	The counter ID

Type	Name	Description
"Counter" on page 88	body	Fields of the counter to update

Table 47. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverCountersCounterGet

Returns the counter spec details of the particular counter.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
Counter DefaultApi#serverCountersCounterGet(String server, String counter)
```

Table 48. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	counter	The counter ID

Table 49. Returns

Type	Notes
"Counter" on page 88	Counter spec details

DefaultApi#serverCountersCounterDelete

Removes the counter specification, similar to the `p4 counter -d` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverCountersCounterDelete(String server, String counter)
```

Table 50. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	counter	The counter ID

Table 51. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverCountersCounterIncrementPost

Increments a numerical counter, similar to the `p4 counter -i` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverCountersCounterIncrementPost(String server, String counter)
```

Table 52. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	counter	The counter ID

Table 53. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverDepotsGet

Lists available depots in the system. The resources of this list are summaries of depots in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<DepotsCommand> DefaultApi#serverDepotsGet(String server)
```

Table 54. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 55. Returns

Type	Notes
List< ["DepotsCommand" on page 90](#) >	Summaries of depots in the system.

DefaultApi#serverDepotsPost

Creates a new depot specification, like the `p4 depot` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverDepotsPost(String server, DepotCommand depot)
```

Table 56. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
["DepotCommand" on page](#)	depot	The depot spec

Table 57. Returns

Type	Notes
["CommandResponse" on page 87](#)	

DefaultApi#serverDepotsDepotGet

Returns the depot spec details of the particular depot.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
DepotCommand DefaultApi#serverDepotsDepotGet(String server, String depot)
```

Table 58. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	depot	The depot ID

Table 59. Returns

Type	Notes
"DepotCommand" on page 88	Depot spec details

DefaultApi#serverDepotsDepotPatch

Update depot specifications, similar to the `p4 depot` command. Only the specified parameters in the body will be changed.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverDepotsDepotPatch(String server, String depot, DepotCommand body)
```

Table 60. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	depot	The depot ID

Type	Name	Description
"DepotCommand" on pa	body	Fields of the depot to update

Table 61. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverDepotsDepotDelete

Removes the depot specification, similar to the `p4 depot -d` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverDepotsDepotDelete(String server, String depot)
```

Table 62. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	depot	The depot ID

Table 63. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverGroupsGet

Lists available groups in the system. The resources of this list are summaries of groups in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<GroupsCommand> DefaultApi#serverGroupsGet(String server)
```

Table 64. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 65. Returns

Type	Notes
List< "GroupsCommand" on page 100 >	Summaries of groups in the system.

DefaultApi#serverGroupsPost

Creates a new group specification, like the **p4 group** command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverGroupsPost(String server, GroupCommand body)
```

Table 66. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
"GroupCommand" on p	body	The group spec

Table 67. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverGroupsGroupGet

Returns the group spec details of the particular group.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
GroupCommand DefaultApi#serverGroupsGroupGet(String server, String group)
```

Table 68. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	group	The group ID

Table 69. Returns

Type	Notes
"GroupCommand" on page 99	Group spec details

DefaultApi#serverGroupsGroupPatch

Update group specifications, similar to the `p4 group` command. Only the specified parameters in the body will be changed.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverGroupsGroupPatch(String server, String group, GroupCommand body)
```

Table 70. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	group	The group ID
"GroupCommand" on p	body	Fields of the group to update

Table 71. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverGroupsGroupDelete

Removes the group specification, similar to the `p4 group -d` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverGroupsGroupDelete(String server, String group)
```

Table 72. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	group	The group ID

Table 73. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverJobsGet

Lists available jobs in the system. The resources of this list are summaries of jobs in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<JobsCommand> DefaultApi#serverJobsGet(String server)
```

Table 74. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 75. Returns

Type	Notes
List< "JobsCommand" on page 101 >	Summaries of jobs in the system.

DefaultApi#serverJobsPost

Creates a new job specification, like the `p4 job` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverJobsPost(String server, JobCommand job)
```

Table 76. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
"JobCommand" on page	job	The job spec

Table 77. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverJobsJobGet

Returns the job spec details of the particular job.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
JobCommand DefaultApi#serverJobsJobGet(String server, String job)
```

Table 78. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	job	The job ID

Table 79. Returns

Type	Notes
[“JobCommand” on page 101](#)	Job spec details

DefaultApi#serverJobsJobPatch

Update job specifications, similar to the `p4 job` command. Only the specified parameters in the body will be changed.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverJobsJobPatch(String server, String job, JobCommand jobCommand)
```

Table 80. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	job	The job ID
[“JobCommand” on page](#)	jobCommand	Fields of the job to update

Table 81. Returns

Type	Notes
[“CommandResponse” on page 87](#)	

DefaultApi#serverJobsJobDelete

Removes the job specification, similar to the `p4 job -d` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverJobsJobDelete(String server, String job)
```

Table 82. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	job	The job ID

Table 83. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverJobsJobFixesChangeDelete

Removes the fix record association for the job for a particular changelist.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverJobsJobFixesChangeDelete(String server, String job, String
  change)
```

Table 84. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	job	The job ID
String	change	The change ID

Table 85. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverJobsJobFixesChangePost

Adds a fix record to the job for a particular changelist.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverJobsJobFixesChangePost(String server, String job, String change,
  String status)
```

Table 86. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	job	The job ID
String	change	The change ID
String	status	Specify the job status instead of using the default. The default is typically **closed** or some other value defined in the **Presets** field specified in the **p4 jobspec** form.
		If the changelist to which you're linking the job been submitted, the status value is immediately reflected in the job's status.
		If the changelist is pending, the job status is changed on submission of the changelist, provided that the -s option is also supplied to p4 submit and the desired status appears next to the job in the p4 submit form's Jobs: field. To leave a job unchanged, use the special status of same.

Table 87. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverLabelsGet

Lists available labels in the system. The resources of this list are summaries of labels in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<LabelsCommand> DefaultApi#serverLabelsGet(String server)
```

Table 88. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 89. Returns

Type	Notes
List< ["LabelsCommand" on page 102](#) >	Summaries of labels in the system.

DefaultApi#serverLabelsPost

Creates a new label specification, like the `p4 label` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverLabelsPost(String server, LabelCommand label)
```

Table 90. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
["LabelCommand" on pa](#)	label	The label spec

Table 91. Returns

Type	Notes
["CommandResponse" on page 87](#)	

DefaultApi#serverLabelsLabelGet

Returns the label spec details of the particular label.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
LabelCommand DefaultApi#serverLabelsLabelGet(String server, String label)
```

Table 92. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	label	The label ID

Table 93. Returns

Type	Notes
"LabelCommand" on page 103	Label spec details

DefaultApi#serverLabelsLabelPatch

Update label specifications, similar to the `p4 label` command. Only the specified parameters in the body will be changed.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverLabelsLabelPatch(String server, String label, LabelCommand
 labelCommand)
```

Table 94. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	label	The label ID
"LabelCommand" on pa	labelCommand	Fields of the label to update

Table 95. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverLabelsLabelDelete

Removes the label specification, similar to the `p4 label -d` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverLabelsLabelDelete(String server, String label)
```

Table 96. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	label	The label ID

Table 97. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverLoginPost

Logs into a Helix Versioning Engine (p4d) server.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
LoginResponse DefaultApi#serverLoginPost(String server, LoginRequest body)
```

Table 98. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
"LoginRequest" on page	body	The user login and password.

Table 99. Returns

Type	Notes
"LoginResponse" on page 105	Object with ticket to use for Basic auth password.

DefaultApi#serverPathsGet

Lists depots, files, and directories in the system. This combines the output of the `p4 depots`, `p4 dirs`, and `p4 files` commands, depending upon your path.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<Location> DefaultApi#serverPathsGet(String server, String path)
```

Table 100. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	path	The path "under a depot" to query under, e.g., `//depot/main`. This will list the directories and files underneath that path.

Table 101. Returns

Type	Notes
List< "Location" on page 104 >	Array of depots.

DefaultApi#serverProtectionsPut

Updates the protections table.

This method requires superuser access.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverProtectionsPut(String server, Protections protections)
```

Table 102. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
"Protections" on page 10	protections	The new protections table

Table 103. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverProtectionsGet

Returns a list of available protections in the system. The elements of this list are rows of the system's protections table.

This method requires superuser access.

See the output of **p4 protect** for more information.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
Protections DefaultApi#serverProtectionsGet(String server)
```

Table 104. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 105. Returns

Type	Notes
"Protections" on page 106	Object including list of protections entries

DefaultApi#serverServersGet

Lists available servers in the system. The resources of this list are summaries of servers in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<ServersCommand> DefaultApi#serverServersGet(String server)
```

Table 106. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 107. Returns

Type	Notes
List< "ServersCommand" on page 107 >	Summaries of servers in the system.

DefaultApi#serverServersPost

Creates a new server specification, like the **p4 server** command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverServersPost(String server, ServerCommand serverCommand)
```

Table 108. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
"ServerCommand" on p:	serverCommand	The server spec

Table 109. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverServersServerIdGet

Returns the server spec details of the particular server.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
ServerCommand DefaultApi#serverServersServerIdGet(String server, String serverId)
```

Table 110. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	serverId	The server ID of the server spec

Table 111. Returns

Type	Notes
"ServerCommand" on page 109	Server spec details

DefaultApi#serverServersServerIdPatch

Update server specifications, similar to the `p4 server` command. Only the specified parameters in the body will be changed.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverServersServerIdPatch(String server, String serverId,
  ServerCommand serverCommand)
```

Table 112. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	serverId	The server ID

Type	Name	Description
"ServerCommand" on pa	serverCommand	Fields of the server to update

Table 113. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverServersServerIdDelete

Removes the server specification, similar to the **p4 server -d** command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverServersServerIdDelete(String server, String serverId)
```

Table 114. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	serverId	The server ID

Table 115. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverStreamsGet

Lists available streams in the system. The resources of this list are summaries of streams in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<StreamsCommand> DefaultApi#serverStreamsGet(String server)
```

Table 116. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 117. Returns

Type	Notes
List< "StreamsCommand" on page 116 >	Summaries of streams in the system.

DefaultApi#serverStreamsPost

Creates a new stream specification, like the `p4 stream` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverStreamsPost(String server, StreamCommand body)
```

Table 118. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
"StreamCommand" on p	body	The stream spec

Table 119. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverStreamsStreamGet

Returns the stream spec details of the particular stream.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
StreamCommand DefaultApi#serverStreamsStreamGet(String server, String stream)
```

Table 120. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	stream	The stream ID

Table 121. Returns

Type	Notes
"StreamCommand" on page 112	Stream spec details

DefaultApi#serverStreamsStreamPatch

Update stream specifications, similar to the `p4 stream` command. Only the specified parameters in the body will be changed.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverStreamsStreamPatch(String server, String stream, StreamCommand
  body)
```

Table 122. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	stream	The stream ID
"StreamCommand" on p	body	Fields of the stream to update

Table 123. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverStreamsStreamDelete

Removes the stream specification, similar to the `p4 stream -d` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverStreamsStreamDelete(String server, String stream)
```

Table 124. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	stream	The stream ID

Table 125. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverTriggersPut

Updates the triggers table.

This method requires superuser access.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverTriggersPut(String server, Triggers triggers)
```

Table 126. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Type	Name	Description
"Triggers" on page 118	triggers	The new triggers table

Table 127. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverTriggersGet

Returns a list of available triggers in the system. The elements of this list are rows of the system's triggers table.

This method requires superuser access.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
Triggers DefaultApi#serverTriggersGet(String server)
```

Table 128. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 129. Returns

Type	Notes
"Triggers" on page 118	List of triggers entries

DefaultApi#serverUsersGet

Lists available users in the system. The resources of this list are summaries of users in the system.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<UsersCommand> DefaultApi#serverUsersGet(String server,  includeService, Integer max)
```

Table 130. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
	includeService	If true, shows service users in the list.
Integer	max	Cap the number of users reported to this amount.

Table 131. Returns

Type	Notes
List< "UsersCommand" on page 120 >	Summaries of users in the system.

DefaultApi#serverUsersPost

Creates a new user specification, like the `p4 user` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverUsersPost(String server, UserCommand body)
```

Table 132. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
"UserCommand" on pag	body	The user spec

Table 133. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverUsersUserGet

Returns the user spec details of the particular user.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
UserCommand DefaultApi#serverUsersUserGet(String server, String user)
```

Table 134. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	user	The user ID

Table 135. Returns

Type	Notes
[“UserCommand” on page 119](#)	User spec details

DefaultApi#serverUsersUserPatch

Update user specifications, similar to the `p4 user` command. Only the specified parameters in the body will be changed.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverUsersUserPatch(String server, String user, UserCommand body)
```

Table 136. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	user	The user ID

Type	Name	Description
"UserCommand" on pag	body	Fields of the user to update

Table 137. Returns

Type	Notes
"CommandResponse" on page 87	

DefaultApi#serverUsersUserDelete

Removes the user specification, similar to the `p4 user -d` command.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse DefaultApi#serverUsersUserDelete(String server, String user)
```

Table 138. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	user	The user ID

Table 139. Returns

Type	Notes
"CommandResponse" on page 87	

AlphaApi Reference

AlphaApi#serverChangesPost

Create a new changelist that can affect multiple files using different kinds of actions. If you require the ability to integrate or move, for example, you can use this method.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse AlphaApi#serverChangesPost(String server, ChangelistRequest changelistRequest)
```

Table 140. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
"ChangelistRequest" on	changelistRequest	Description of changes to make

Table 141. Returns

Type	Notes
"CommandResponse" on page 87	

AlphaApi#serverGitFusionReposGet

Lists all configured repositories readable by the current user. .Package Name

```
com.perforce.helix_web_services_client
```

Method Signature.

```
List<GitFusionRepoId> AlphaApi#serverGitFusionReposGet(String server)
```

Table 142. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Table 143. Returns

Type	Notes
List< "GitFusionRepoId" on page 94 >	List of configured repository names and IDs

AlphaApi#serverGitFusionReposPost

Submits a [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file to create or update a repository configuration.

If the repository does not exist or has been previously deleted, this method saves contents of the config file to a new **p4gf_config** file. If the repository has already been initialised, this method replaces all of the file contents of the specified repository's **p4gf_config** file.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse AlphaApi#serverGitFusionReposPost(String server, GitFusionRepoConfig body)
```

Table 144. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
"GitFusionRepoConfig"	body	The new configuration

Table 145. Returns

Type	Notes
"CommandResponse" on page 87	

AlphaApi#serverGitFusionReposRepoGet

Return configuration for the specified repository. Grabs and returns contents of the [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file for given repository.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
GitFusionRepoConfig AlphaApi#serverGitFusionReposRepoGet(String server, String repo)
```

Table 146. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.

Type	Name	Description
String	repo	The Git Fusion Repo ID

Table 147. Returns

Type	Notes
"GitFusionRepoConfig" on page	Git Fusion repository config

AlphaApi#serverGitFusionReposRepoPatch

Updates values in the repository configuration. This method will find all specified parameters and update each value for the specified repository's configuration file.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse AlphaApi#serverGitFusionReposRepoPatch(String server, String repo,
 GitFusionRepoConfig body)
```

Table 148. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	repo	The Git Fusion Repo ID
"GitFusionRepoConfig"	body	The new configuration

Table 149. Returns

Type	Notes
"CommandResponse" on page 87	

AlphaApi#serverGitFusionReposRepoDelete

Deletes the repository configuration (The [p4gf_config file](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j)). Contents of the repository are not deleted from Perforce.

Package Name.

```
com.perforce.helix_web_services_client
```

Method Signature.

```
CommandResponse AlphaApi#serverGitFusionReposRepoDelete(String server, String repo)
```

Table 150. Parameters

Type	Name	Description
String	server	The server ID that we execute this particular method against.
String	repo	The Git Fusion Repo ID

Table 151. Returns

Type	Notes
"CommandResponse" on page 87	

Java Models

Each Java model provides getter and setter accessor methods to each listed property name.

BranchCommand

Models the output of a `p4 branch` command.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_branch.html).

Table 152. Properties

Type	Name	Description
String	branch	The branch name, as provided on the command line.
String	owner	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.
java.util.Date	access	The date the branch mapping was last accessed.
java.util.Date	update	The date the branch mapping was last changed.

Type	Name	Description
String	options	Either `unlocked` (the default) or `locked`. If `locked`, only the `Owner:` can modify the branch mapping, and the mapping can't be deleted until it is `unlocked`.
String	description	A short description of the branch's purpose.
List<String>	view	A set of mappings from one set of files in the depot (the source files) to another set of files in the depot (the target files). The view maps from one location in the depot to another; it can't refer to a client workspace. `For example, the branch view `//depot/` `main/... //depot/r2.1/...` maps all the files` ` under `//depot/main` to `//depot/r2.1`.`

BranchesCommand

A reference to a branch mapping known to the system.

Table 153. Properties

Type	Name	Description
String	branch	The branch name, as provided on the command line.
String	owner	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.
java.util.Date	access	The date the branch mapping was last accessed.
java.util.Date	update	The date the branch mapping was last changed.
String	options	Either `unlocked` (the default) or `locked`. If `locked`, only the `Owner:` can modify the branch mapping, and the mapping can't be deleted until it is `unlocked`.
String	description	A short description of the branch's purpose.

ChangeCommand

A changelist specification.

For more information see the [command reference](https://www.perforce.com/perforce/ [doc.current/manuals/cmdref/p4_change.html](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_change.html)).

Table 154. Properties

Type	Name	Description
String	change	Contains the changelist number if editing an existing changelist, or `new` if creating a new changelist.
String	client	Name of current client workspace
java.util.Date	date	Date the changelist was last modified.
String	user	Name of the change owner. The owner of an empty pending changelist (that is, a pending changelist without any files in it) can transfer ownership of the changelist to another existing user either by editing this field, or by using the -U user option. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
String	status	`pending`, `shelved`, `submitted`, or `new`. Not editable by the user. The status is `new` when the changelist is created, `pending` when it has been created but has not yet been submitted to the depot, `shelved` when its contents are shelved, and `submitted` when its contents have been stored in the depot.
String	description	Textual description of changelist. If you do not have access to a restricted changelist, the description is replaced with a "no permission" message.
List<String>	jobs	A list of jobs that are fixed by this changelist.
String	type	Type of change: `restricted` or `public`. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status:

Type	Name	Description
		field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change.
		Public changes are displayed without restrictions.
		By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.
List<String>	files	The list of files submitted in this changelist.
String	importedBy	Displays the name of the user who ran the p4 fetch, p4 push, or p4 unzip command that imported this change into the server.
		This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied.
		In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.
String	identify	Contains a label which uniquely identifies this changelist across all servers where it has been fetched, pushed, or unzipped.
		This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied.
		In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.

Helix Web Services Guide

ChangesCommand

Table 155. Properties

Type	Name	Description
String	change	The changelist ID
java.util.Date	date	Last modification time of the changelist
String	user	The owner of the changelist
String	client	Name of current client workspace.
String	status	`pending`, `shelved`, `submitted`, or `new`. Not editable by the user. The status is `new` when the changelist is created, `pending` when it has been created but has not yet been submitted to the depot, `shelved` when its contents are shelved, and `submitted` when its contents have been stored in the depot.
String	type	Type of change: `restricted` or `public`. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change. Public changes are displayed without restrictions. By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.
String	path	Depot paths affected by this changelist
String	description	Textual description of changelist. If you do not have access to a restricted changelist, the description is replaced with a "no permission" message.

ChangelistRequest

Table 156. Properties

Type	Name	Description
String	description	
String	stream	Optional stream ID to use in case you want to edit files in a stream.
List<"ChangelistAction"	actions	

ChangelistAction

Table 157. Properties

Type	Name	Description
String	depotFile	The target file path to edit.
String	fromDepotFile	For "branch" or "move" actions, this indicates the source file location.
String	actionType	One of "upload", "branch", "move", or "delete"
String	content	Base64-encoded content
Integer	requireVersion	If set, we will only operate if this is the current version of the file.

ClientCommand

The client workspace specification and its view.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html).

Table 158. Properties

Type	Name	Description
String	client	The client workspace name, as specified in the P4CLIENT environment variable or its equivalents.
String	owner	The name of the user who owns the workspace. The default is the user who created the workspace. The specified owner does not have to be a Perforce user. You might want to use an

Type	Name	Description
		arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
java.util.Date	update	The time the workspace specification was last modified.
java.util.Date	access	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)
String	host	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host. The hostname must be provided exactly as it appears in the output of p4 info when run from that host. This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.
String	description	A textual description of the workspace. The default text is Created by owner.
String	root	The directory (on the local host) relative to which all the files in the `View:` are specified. The default is the current working directory. The path must be specified in local file system syntax. If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to map files to multiple drives.
List<String>	altRoots	Up to two optional alternate client workspace roots. Perforce applications use the first of the main and alternate roots that match the application's

Type	Name	Description
		current working directory. Use the p4 info command to display the root being used. This enables users to use the same Perforce client workspace specification on multiple platforms, even those with different directory naming conventions. If you are using multiple or alternate workspace roots (the AltRoots: field), you can always tell which root is in effect by looking at the Client root: reported by p4 info. If you are using a Windows directory in any of your workspace roots, you must specify the Windows directory as your main workspace root and specify your other workspace roots in the AltRoots: field.
String	options	A set of seven switches that control particular workspace options. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.

String	submitOptions	Options to govern the default behavior of p4 submit.
		
		• submitunchanged
		
		  All open files (with or without changes) are submitted to the depot. This is the default behavior of Perforce.
		
		• submitunchanged+reopen
		
		  All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist.
		
		• revertunchanged
		
		  Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted.
		
		• revertunchanged+reopen

Type	Name	Description
		Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist.
		• leaveunchanged
		Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist.
		• leaveunchanged+reopen
		Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged+reopen, except that no unchanged files are submitted to the depot.
String	lineEnd	Configure carriage-return/linefeed (CR/LF) conversion. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
String	stream	Associates the workspace with the specified stream. Perforce generates the view for stream-associated workspaces: you cannot modify it manually.
String	streamAtChange	A changelist number that sets a back-in-time view of a stream. When StreamAtChange is set, running p4 sync (when called with no arguments) updates the workspace to files at this changelist revision, instead of the head revision. You cannot submit changes (p4 submit returns an error) when StreamAtChange is set, because the workspace view no longer reflects the current stream inheritance.

Type	Name	Description
		This field is ignored unless the Stream field is also set to a valid stream.
String	serverID	If set, restricts usage of the workspace to the named server. If unset, use is allowed on master server and on any replicas of the master other than Edge servers.
List<String>	view	Specifies the mappings between files in the depot and files in the workspace. A new view takes effect on the next p4 sync operation.
List<String>	changeView	Restricts access to depot paths to a particular point in time. Files specified for the ChangeView field are read-only: they may be opened but not submitted. For example: **//depot/path/…@1000** Revisions of the files in the specified path will not be visible if they were submitted after the specified changelist number. Files matching a ChangeView path may not be submitted.
String	type	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts. Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.

ClientsCommand

Table 159. Properties

Type	Name	Description
String	client	The client workspace name, as specified in the **P4CLIENT** environment variable or its equivalents.

Type	Name	Description
String	owner	The name of the user who owns the workspace. The default is the user who created the workspace.
		The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
java.util.Date	update	The time the workspace specification was last modified.
java.util.Date	access	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)
String	host	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host.
		The hostname must be provided exactly as it appears in the output of p4 info when run from that host.
		This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.
String	description	A textual description of the workspace. The default text is Created by owner.
String	root	The directory (on the local host) relative to which all the files in the View: are specified. The default is the current working directory. The path must be specified in local file system syntax.
		If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the

Type	Name	Description
		root as null to enable you to map files to multiple drives. additionalProperties:
String	type	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts. Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.
String	options	A set of seven switches that control particular workspace options. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
String	submitOptions	Options to govern the default behavior of p4 submit. • submitunchanged All open files (with or without changes) are submitted to the depot. This is the default behavior of Perforce. • submitunchanged+reopen All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist. • revertunchanged Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted. • revertunchanged+reopen

Type	Name	Description
		Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist.
		• leaveunchanged
		Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist.
		• leaveunchanged+reopen
		Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged+reopen, except that no unchanged files are submitted to the depot.
String	lineEnd	Configure carriage-return/linefeed (CR/LF) conversion. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
String	stream	Associates the workspace with the specified stream. Perforce generates the view for stream-associated workspaces: you cannot modify it manually.

CommandResponse

A generic container for responses from the p4d server that we have yet to completely classify.

Table 160. Properties

Type	Name	Description
List<object>	results	A collection of maps that have various values set by p4d.

CommandRequest

A single map typically defines input to generic command methods.

Table 161. Properties

Type	Name	Description
Object	object	Don't use this. It's a kludge around a bug in the Java client code generator

Counter

A persistent variable in the server.

Table 162. Properties

Type	Name	Description
String	counter	The variable name
String	value	The variable value. Many variables are numerical in nature, which allow you to do atomic increment operations in method calls instead of having to fetch, increment, and save.

DepotCommand

The depot specification, which is the shared repository Perforce stores files in.

Table 163. Properties

Type	Name	Description
String	depot	The depot name.
String	owner	The user who owns the depot. By default, this is the user who created the depot. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
String	description	A short description of the depot's purpose. Optional.
String	type	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`.

Type	Name	Description
		A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server.
		A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy.
		A remote depot references files that reside on other servers, and cannot be written to.
		The spec depot, if present, automatically archives edited forms.
		The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.
		An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries.
		A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.
String	address	If the `Type:` is `remote`, the address should be the P4PORT address of the remote server. If the `Type:` is local or spec, this field is ignored.
String	suffix	If the Type: is spec, this field holds an optional suffix for generated paths to objects in the spec depot.
		The default suffix is .p4s. You do not need a suffix to use the spec depot, but supplying a file extension to your Perforce server's versioned specs enables users of GUI client software to associate Perforce specifications with a preferred text editor. If the Type: is local or remote, this field is ignored.
String	streamDepth	For stream depots, the optional depth to be used for stream paths in the depot, where depth

Type	Name	Description
		specifies the number of slashes following the depot name of a stream.] This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.
String	map	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the … wildcard; for example, a local depot new might have a Map: of `new/…`. If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, `//depot/new/rel2/…`. This directory will be the root of the local representation of the remote depot.
List<String>	specMap	For spec depots, an optional description of which specs should be saved, expressed as a view.

DepotsCommand

A summary of depots in the system, with information provided by the `p4 depots` command.

Table 164. Properties

Type	Name	Description
String	depot	The depot name.
String	map	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the … wildcard; for example, a local depot new might have a Map: of `new/…`. If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, `//depot/new/rel2/…`. This directory will be the root of the local representation of the remote depot.
String	type	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`.

Type	Name	Description
		A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server.
		A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy.
		A remote depot references files that reside on other servers, and cannot be written to.
		The spec depot, if present, automatically archives edited forms.
		The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.
		An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries.
		A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.
String	streamDepth	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following the depot name of a stream.]
		This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.
String	description	A short description of the depot's purpose. Optional.

DirsCommand

Table 165. Properties

Type	Name	Description
String	dir	

FilesCommand

Table 166. Properties

Type	Name	Description
String	depotFile	
String	revision	
String	change	
String	action	
java.util.Date	time	
String	type	

FstatCommand

Detailed information about each file, as provided by the `p4 fstat` command.

Table 167. Properties

Type	Name	Description
String	depotFile	Depot path to file. For files containing special characters, the filename is displayed containing the ASCII expression of the character's hexadecimal value.
String	movedFile	Name in depot of moved to/from file.
String	shelved	Set to `shelved` if file is shelved.
String	headAction	Action taken at head revision, if in depot. One of: add, edit, delete, branch, move/add, move/delete, integrate, import, purge, or archive.
String	headChange	Head revision changelist number, if in depot.
String	headRev	Head revision number, if in depot.

Type	Name	Description
String	headType	Head revision type, if in depot.
String	headCharset	Head charset, for unicode files.
java.util.Date	headTime	Head revision changelist time, if in depot. Time is measured in seconds since 00:00:00 UTC, January 1, 1970.
java.util.Date	headModTime	Head revision modification time (the time that the file was last modified on the client before submit), if in depot.
String	movedRev	Head revision of moved file.
String	digest	MD5 digest of a file.
String	fileSize	File length in bytes.
String	actionOwner	User who opened the file, if open.
String	resolved	The number, if any, of resovled integration records.
String	unresolved	The number, if any, of unresolved integration records.
String	reresolvable	The number, if any, of re-resolvable integration records.
List<String>	otherOpens	For each user with the file open, the workspace and user with the open file.
List<String>	otherLocks	For each user with the file locked, the workspace and user holding the lock.
List<String>	otherActions	For each user with the file open, the action taken.
List<String>	otherChanges	The changelist number with this file open.
List<String>	resolveActions	Pending integration action.
List<String>	resolveBaseFiles	Pending base files.
List<String>	resolveBaseRevs	Pending base revision numbers.
List<String>	resolveFromFiles	Pending from files.
List<String>	resolveStartFromRevs	Pending starting revisions.
List<String>	resolveEndFromRevs	Pending ending revisions.

GitFusionRepoId

Table 168. Properties

Type	Name	Description
String	id	An identifier for the repository that can be used safely within URL paths.
String	name	The repository name, which can be path-like.

GitFusionRepoConfig

Table 169. Properties

Type	Name	Description
String	name	The repository name, which can be path-like.
String	description	Repo description returned by the @list command.
"GitFusionRepoGlobalO	globalOverrides	
List<"GitFusionRepoBra	branches	

GitFusionRepoBranchConfig

Defines a unique Git Fusion branch.

Table 170. Properties

Type	Name	Description
String	gitBranchId	Alphanumeric ID for the git branch. *Do not change this value once this repo has been cloned*.
String	gitBranchName	Defines a name specified in a local repo for a Git branch. A valid Git branch name. Do not edit this value after you clone the repo.
List<String>	view	Defines a Perforce workspace view mapping that maps Perforce depot paths (left side) to Git work tree paths (right side). Correctly formed mapping syntax; must not include any Perforce stream or spec depots, and all depot paths on the right side must match

Type	Name	Description
		exactly across all branch definitions. You can add and remove only certain types of Perforce branches from this view after you clone the repo.
String	stream	Defines a Perforce stream that maps to the Git branch. Provide a stream name using the syntax // streamdepot/mystream. A Git Fusion branch can be defined as a view or a stream but not both. If your branch is defined as stream, it can include only one stream.
String	readOnly	Prohibit git pushes that introduce commits to the branch.

GitFusionRepoGlobalOverrides

A list of per-repo settings that override global settings.

Table 171. Properties

Type	Name	Description
String	charset	Defines the default Unicode setting that Git Fusion applies to new repos. This setting is valid only when Git Fusion interacts with a Unicode-enabled Perforce server. (Defaults to `UTF-8`).
String	depotPathRepoCreationl	Allow Git users to create new repos by pushing/pulling a git url which specifies a Perforce depot path. This is similar to creating a repo from a p4 client.
String	depotPathRepoCreationl	Restrict which authenticated Git pushers are allowed to create new repos when depot-path-repo-creation-enable is enabled.
String	changeOwner	Defines whether Git Fusion assigns either the Git commit author or the Git pusher as the owner of a pushed change (submit).
String	enableGitBranchCreatior	Defines whether Git Fusion creates a new branch of Perforce depot file hierarchy for each copied branch of Git workspace history, including Git task branches as Git Fusion anonymous branches.

Type	Name	Description
String	enableSwarmReviews	Permits branch creation for Swarm reviews, even when enable-git-branch-creation is disabled.
String	enableGitMergeCommits	Defines whether Git Fusion copies merge commits and displays them in Perforce as integrations between Perforce branches.
String	enableGitSubmodules	Defines whether Git Fusion allows Git submodules to be pushed to Perforce.
String	ignoreAuthorPermission	Defines whether Git Fusion evaluates both the author's and pusher's Perforce write permissions during a push or evaluates only the pusher's permissions.
String	preflightCommit	Enables you to trigger pre-flight commit scripts that enforce local policy for Git pushes. This can be especially useful if you have Perforce submit triggers that could reject a push and damage the repository.
String	readPermissionCheck	Enables you to require that Git clone, pull, or fetch requests check the Perforce protections table for the puller's read permission on the files being pulled.
String	gitMergeAvoidanceAfter	If the Perforce service includes any changelists submitted by Git Fusion 13.2 or earlier, you can prevent unnecessary merge commits by setting this key to the number of the last changelist submitted before your site upgraded to a later version of Git Fusion.
String	jobLookup	Set the format for entering Perforce jobs in Git commit descriptions so that they are recognized by Git Fusion and appear in Perforce changelists as fixes. By default, job IDs whose string starts with "job" (as in job123456) are passed through to the changelist description and job field. Use this option if you want Git Fusion to recognize additional expressions, such as JIRA issue IDs.
String	depotBranchCreationEna	Allow Git users to create new fully-populated depot branches within Perforce.
String	depotBranchCreationP4g	Restrict the authenticated Git pushers who are allowed to create new fully-populated depot branches, if depotBranchCreationEnable is enabled.

Type	Name	Description
String	depotBranchCreationDep	Tell Git Fusion where to create new fully-populated depot branches, if depotBranchCreationEnable is enabled.
		Default path is **//depot/[repo]/ [git_branch_name]**.
String	depotBranchCreationVie	Set how the depot path set in depotBranchCreationDepotPath should appear in Git.
		Enter a Perforce view specification that maps Perforce depot paths (left side) to Git work tree paths (right side). Perforce depot paths are relative to the root set in depotBranchCreationDepotPath.
		The default … … maps every file under the depotBranchCreationDepotPath root to Git. Right side paths must match the right side for every other branch already defined within a repo.
String	enableGitFindCopies	When Git reports a copy file action, store that action in Perforce as a p4 integ. Often set in tandem with enableGitFindRenames.
		No/Off/0%: Do not use Git's copy detection. Treat all possible file copy actions as p4 add actions.
		1%-100%: Use Git's copy detection. Value passed to git diff-tree --find-copies=n.
		Git Fusion also adds --find-copies-harder whenever adding --find-copies.
String	enableGitFindRenames	When Git reports a rename (also called move) file action, store that in Perforce as a p4 move. Often set in tandem with enableGitFindCopies.
		No/**Off**/**0%**: Do not use Git's rename detection. Treat all possible file rename actions as independent p4 delete and p4 add actions.
		1%-100%: Use Git's rename detection. Value passed to git diff-tree --find-renames=n.
String	enableStreamImports	Enables you to convert Perforce stream import paths to Git submodules when you clone a Git

Type	Name	Description
		Fusion repository. If set to Yes, you must also set either httpUrl or sshUrl.
String	httpUrl	The URL used by Git to clone a repository from Git Fusion over HTTP. This property is required if you want to use Perforce stream import paths as git submodules and you use HTTP(S).
String	sshUrl	The "URL" used by Git to clone a repository from Git Fusion using SSH. This property is required if you want to use Perforce stream import paths as git submodules and you use SSH.
String	emailCaseSensitivity	Defines whether Git Fusion pays attention to case when matching Git user email addresses to Perforce user account email addresses during the authorization check.
String	authorSource	Defines the source that Git Fusion uses to identify the Perforce user associated with a Git push. Defaults to `git-email`. Use any one of the following values: • `git-email`: Use the email address of the Git author to look for a Perforce user account with the same email address. Git Fusion consults the p4gf_usermap file first, and if that fails to produce a match, it scans the Perforce user table. • `git-user`: Use the user.name field in the Git commit. This is the part of the author field before the email address. • `git-email-account`: Use the account portion of the Git author's email address. If the Git author's email value is `samwise@the_shire.com`, Git Fusion uses the Perforce account samwise. You can also tell Git Fusion to iterate through multiple source types until it finds a matching Perforce account. Specify the source types in order of precedence, separated by commas. For example: git-user, git-email-account, git-email.

Type	Name	Description
String	limitSpaceMb	Natural number representing the number of megabytes of disk space that can be consumed by any single repo. This value does not include the spaced consumed on the Perforce server.
String	limitCommitsReceived	Natural number representing the maximum number of commits allowed in a single push.
String	limitFilesReceived	Natural number representing the maximum number of files allowed in a single push.
String	limitMegabytesReceived	Natural number representing the maximum number of megabytes allowed in a single push.

GroupCommand

Add or delete users from a group, or set the maxresults, maxscanrows, maxlocktime, and timeout limits for the members of a group.

Table 172. Properties

Type	Name	Description
String	group	The name of the group, as entered on the command line.
String	maxResults	The maximum number of results that members of this group can access from the service from a single command. The default value is **unset**.
String	maxScanRows	The maximum number of rows that members of this group can scan from the service from a single command. The default value is **unset**.
String	maxLockTime	The maximum length of time (in milliseconds) that any one operation can lock any database table when scanning data. The default value is **unset**.
String	maxOpenFiles	The maximum number of files that a member of a group can open using a single command.
String	timeout	The duration (in seconds) of the validity of a session ticket created by p4 login. The default value is 43,200 seconds (12 hours). To create a ticket that does not expire, set the Timeout: field to **unlimited**.

Type	Name	Description
String	passwordTimeout	The length of time (in seconds) for which passwords for users in this group remain valid. To disable password aging, use a value of unset.
String	ldapConfig	The LDAP configuration to use when populating the group's user list from an LDAP query.
String	ldapSearchQuery	The LDAP query used to identify the members of the group.
String	ldapUserAttribute	The LDAP attribute that represents the user's username.
List<String>	subgroups	Names of other Perforce groups. To add all users in a previously defined group to the group you're presently working with, include the group name in the Subgroups: field of the p4 group form. Note that user and group names occupy separate namespaces, and thus, groups and users can have the same names. Every member of any previously defined group you list in the Subgroups: field will be a member of the group you're now defining.

List<String>	owners	Names of other Perforce users.
		
		`Group owners without super access are permitted to administer this group, provided that they use the -a option.`
		
		`Group owners are not necessarily members of a group; if a group owner is to be a member of the group, the userid must also be added to the Users: field.`
		
		`The specified owner does not have to be a Perforce user.`
		`You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.`

List<String>	users	The Perforce usernames of the group members.

GroupsCommand

A list of entries that can show the layout how users are associated with the different groups in the system.

Table 173. Properties

Type	Name	Description
String	user	
String	group	
String	isSubGroup	
String	isOwner	
String	isUser	
String	maxResults	
String	maxScanRows	
String	maxLockTime	
String	maxOpenFiles	
String	timeout	
String	passTimeout	

HWSStatus

Table 174. Properties

Type	Name	Description
String	status	When "OK" the server should be considered to be operating normally
String	version	The version of Helix Web Services server.

JobCommand

A defect, enhancement request, or other job specification.

The actual fields in a job can be edited by a superuser in your system. The default set of fields in a system are Job, Status, User, Date, and Description.

Table 175. Properties

Type	Name	Description
String	Job	The job name.

JobsCommand

A summary of jobs in the system.

The actual fields in a job can be edited by a superuser in your system. The default set of fields in a system are Job, Status, User, Date, and Description. Fields in the output of this command may be missing if the superuser removed User, Status, Date, or Description.

Table 176. Properties

Type	Name	Description
String	Job	The job name.

LabelsCommand

Table 177. Properties

Type	Name	Description
String	label	The label name.
java.util.Date	update	The date the label specification was last modified.
java.util.Date	access	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)
String	owner	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
String	options	Options to control behavior and storage location of labels. • locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync. • autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload

Type	Name	Description
		depot can improve performance on sites that make extremely heavy use of labels.
String	description	An optional description of the label's purpose.

LabelCommand

A label specification.

Labels can be either automatic or static. Automatic labels refer to the revisions provided in the View: and Revision: fields. Static labels refer only to those specific revisions tagged by the label by means of either the p4 labelsync or p4 tag commands.

Table 178. Properties

Type	Name	Description
String	label	The label name.
String	owner	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
java.util.Date	update	The date the label specification was last modified.
java.util.Date	access	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)
String	description	An optional description of the label's purpose.
String	options	Options to control behavior and storage location of labels. • locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync. • autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in

Type	Name	Description
		db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels.
String	revision	An optional revision specification for an automatic label. If you use the # character to specify a revision number, you must use quotes around it in order to ensure that the # is parsed as a revision specifier, and not as a comment field in the form.
List<String>	view	A list of depot files that can be tagged with this label. No files are actually tagged until `p4 labelsync` is invoked. Unlike client views or branch views, which map one set of files to another, label views consist of a simple list of depot files.
String	serverID	If set, restricts usage of the label to the named server. If unset, this label may be used on any server.

Location

A consolidated mechanism for identifying something that generally has a path in the system.

Each location references either a depot, a dir, or a file.

Table 179. Properties

Type	Name	Description
String	depotPath	An absolute depot path specification.
"DepotsCommand" on p	depot	
"DirsCommand" on pag	dir	
"FilesCommand" on pag	file	
"FstatCommand" on pag	fstat	
String	content	If this location indicates a single file, this can be set with the Base64-encoded content of the file.

LoginRequest

Captures the login information we need for logging into either a p4d server or our "authentication source".

Table 180. Properties

Type	Name	Description
String	user	Usually the Perforce username
String	password	
List<"ServerLoginReque	serverLogins	

ServerLoginRequest

Table 181. Properties

Type	Name	Description
String	id	The server's ID
String	user	
String	password	

LoginResponse

Either of our login methods return a ticket, which is then used as a password in a basic authentication scheme.

When this is returned from the explicit p4d login, this is a host unlocked ticket, acceptable for using with a local client.

Table 182. Properties

Type	Name	Description
String	ticket	

P4dConfigId

Identification of servers the Helix Web Services instance can connect to.

Table 183. Properties

Type	Name	Description
String	id	A simple string identifier (alphanumeric characters only, please)

Type	Name	Description
String	name	A display string, not guaranteed to be unique
String	description	A simple textual description, for potential selection by clients.

Protections

Displays the information stored in the `p4 protect` command.

Table 184. Properties

Type	Name	Description
List<String>	protections	Each item in the protections array is a line in the protections table, and is split into five columns.
		1. Access level or mode. One of the access levels list, read, open, write, admin, super, review; or one of the rights =read, =open, =write, and =branch,
		2. Either **user** or **group**, to indicate what's identified by this entry.
		3. The group name or user name. To grant permission to all users, use a wildcard with just an asterix symbol.
		4. The IP address of the client host.
		5. The depot file path, which can contain wildcards. To exclude this mapping from the permission set, use a dash - as the first character of this value.
		IPv6 addresses and IPv4 addresses are also supported. You can use the * wildcard to refer to all IP addresses, but only when you are not using CIDR notation.
		If you use the * wildcard with an IPv6 address, you must enclose the entire IPv6 address in square brackets. For example, [2001:db8:1:2:*] is equivalent to [2001:db8:1:2::]/64. Best practice is to use CIDR notation, surround IPv6 addresses with brackets, and to avoid the * wildcard.
		How the system forms host addresses depends on the setting of the dm.proxy.protects variable. By default, this variable is set to 1. This means

Type	Name	Description
		that if the client host uses some intermediary (proxy, broker, replica) to access the server, the proxy- prefix is prepended to the client host address to indicate that the connection is not direct. If you specify proxy-* for the Host field, that will affect all connections made via proxies, brokers, and replicas. A value like proxy-10.0.0.5 identifies a client machine with an IP address of 10.0.0.5 that is connected to the server through an intermediary. Setting the dm.proxy.protects variable to 0, removes the proxy- prefix and allows you to write a single set of protection entries that apply both to directly-connected clients as well as to those that connect via an intermediary. This is more convenient but less secure if it matters that a connection is made using an intermediary. If you use this setting, all intermediaries must be at release 2012.1 or higher.

ServersCommand

Table 185. Properties

Type	Name	Description
String	serverID	A unique identifier for this server. This must match the contents of the server's `server.id` file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.
String	type	Server executable type. One of the following: `server`, `proxy`, `broker`, `identifier`, `admin`. Each type may offer one or more services, defined in the `services` property.
String	services	The `server` type server provides the following services: • standard - a standard Perforce server • replica - a read-only replica server • commit-server - central server in distributed installation

Type	Name	Description
		• edge-server - node in distributed installation
		• forwarding-replica - a replica configured to forward commands that involve database writes to a master server
		• build-server - a replica that supports build automation and build farm integration
		• P4AUTH - a server that provides authentication
		• P4CHANGE - a server that provides change numbering
		• depot-master - commit-server with automated failover
		• depot-standby - standby replica of the depot-master
		• workspace-server - node in a cluster installation
		• standby - read-only replica server that uses p4 journalcopy
		• forwarding-standby - forwarding replica server that uses p4 journalcopy
		The `proxy` type server provides a p4p caching proxy.
		The `broker` type server provides the following services:
		• broker - a p4broker process
		• workspace-router - routing broker for a cluster
		The services field for the `identifier` type server specifies the existence of the cluster, and has the value `cluster`. The name of the cluster is then drawn from the ServerID field.
		The `admin` type server provides the following services:
		• hxca-server - the admin server for a Helix cluster.

Type	Name	Description
		• zookeeper-server - ZooKeeper server for a cluster
String	name	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.
String	address	The P4PORT used by this server.
String	description	An optional description for this server.
String	user	The service user name used by the server.

ServerCommand

The Perforce server specification describes the high-level configuration and intended usage of a Perforce server. For installations with only one Perforce server, the server specification is optional.

Table 186. Properties

Type	Name	Description
String	serverID	A unique identifier for this server. This must match the contents of the server's `server.id` file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.
String	type	Server executable type. One of the following: `server`, `proxy`, `broker`, `identifier`, `admin`. Each type may offer one or more services, defined in the `services` property.
String	services	The `server` type server provides the following services: • standard - a standard Perforce server • replica - a read-only replica server • commit-server - central server in distributed installation • edge-server - node in distributed installation • forwarding-replica - a replica configured to forward commands that involve database writes to a master server

Type	Name	Description
		• build-server - a replica that supports build automation and build farm integration
		• P4AUTH - a server that provides authentication
		• P4CHANGE - a server that provides change numbering
		• depot-master - commit-server with automated failover
		• depot-standby - standby replica of the depot-master
		• workspace-server - node in a cluster installation
		• standby - read-only replica server that uses p4 journalcopy
		• forwarding-standby - forwarding replica server that uses p4 journalcopy
		The `proxy` type server provides a p4p caching proxy.
		The `broker` type server provides the following services:
		• broker - a p4broker process
		• workspace-router - routing broker for a cluster
		The services field for the `identifier` type server specifies the existence of the cluster, and has the value `cluster`. The name of the cluster is then drawn from the ServerID field.
		The `admin` type server provides the following services:
		• hxca-server - the admin server for a Helix cluster.
		• zookeeper-server - ZooKeeper server for a cluster
String	name	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.

Type	Name	Description
String	address	The P4PORT used by this server.
String	externalAddress	For an edge server, this optional field specifies the external address used for connections to a commit server. This field must be set for the edge server to enable parallel submits in a federated environment.
String	description	An optional description for this server.
String	user	The service user name used by the server.
String	clientDataFilter	For a replica server, this optional field can contain one or more patterns describing how active client workspace metadata is to be filtered. Active client workspace data includes have lists, working records, and pending resolves. To include client data, use the syntax: **//client-pattern/…** To exclude client data, use the syntax: **-//client-pattern/…** All patterns are specified in client syntax.
String	revisionDataFilter	For a replica server, this optional field can contain one or more patterns describing how submitted revision metadata is to be filtered. Submitted revision data includes revision records, integration records, label contents, and the files listed in submitted changelists. To include depot data, use the syntax: To exclude depot data, use the syntax: -//depot/pattern/… All patterns are specified in depot syntax.
String	archiveDataFilter	For a replica server, this optional field can contain one or more patterns describing the policy for automatically scheduling the replication of file content. If this field is present, only those files described by the pattern are automatically transferred to the replica; other files are not transferred until they are referenced by a replica command that needs the file content.

Type	Name	Description
		Files specified in the ArchiveDataFilter: field are transferred to the replica regardless of whether any users of the replica have made requests for their content.
		To automatically transfer files on submit, use the syntax: **//depot/pattern/…**
		To exclude files from automatic transfer, use the syntax: **-//depot/pattern/…**
		All patterns are specified in depot syntax.
String	distributedConfig	For an edge or commit server, this optional field, which is displayed only when you use the -l or -c option, shows configuration settings for this server.
		-l flag shows the current configuration. **-c-** flag shows current configuration values, recommended default values for fields that are not set, or unset for fields that are not set and do not have default values.
		If this field is present when invoked with -c, the configuration commands in this field are run on the current server using the scope of the server specified in the serverID field.

StreamCommand

The Perforce stream specification defines a single stream.

Streams are hierarchical branches with policies that control the structure and the flow of change. Stream hierarchies are based on the stability of the streams, specified by the type you assign to the stream. Development streams are least stable (most subject to change), mainline streams are somewhat stable, and release streams are highly stable. Virtual streams can be used to copy and merge between parent and child streams without storing local data. Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data.

Stream contents are defined by the paths that you map. By default, a stream has the same structure as its parent (the stream from which it was branched), but you can override the structure, for example to ensure that specified files cannot be submitted or integrated to other streams.

Table 187. Properties

Type	Name	Description
String	stream	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form **//depotname/streamname**.
java.util.Date	update	The date the stream specification was last modified.
java.util.Date	access	The date and time that the stream specification was last accessed by any Perforce command.
String	owner	The Perforce user or group who owns the stream. The default is the user who created the stream.
String	name	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname portion of the stream path.
String	parent	The parent of this stream. Must be none if the stream's **Type:** is mainline, otherwise must be set to an existing stream identifier of the form **//depotname/streamname**.
String	type	The stream's type determines the expected flow of change. Valid stream types are **mainline**, **virtual**, **development**, and **release**. • **mainline**: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream. • **virtual**: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream. • **release**: More stable than the mainline. Release streams copy from the parent and merge to the parent. • **development**: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent.

Type	Name	Description
		• `task`: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.
String	description	Description of the stream.
String	options	Settings that configure stream behavior as follows: • `[un]locked`: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification cannot be deleted, and only its owner can modify it. The default is unlocked. • `[all,owner]submit`: Specifies whether all users or only the owner of the stream can submit changes to the stream. The default is allsubmit. If the Owner: of a stream marked ownersubmit is a group, all users who are members of that group can submit changes to the stream. • `[no]toparent`: Specifies whether integrations from the stream to its parent are expected. The default is toparent. • `[no]fromparent`: Specifies whether integrations to the stream from its parent are expected. The default is fromparent for mainline and development streams, and nofromparent for release streams. • `mergeany,mergedown`: Specifies whether the merge flow is restricted or whether merge is permitted from any other stream. For example, the mergeany option would allow a merge from a child to a parent with no warnings. A virtual stream must have its flow options set

Type	Name	Description
		to notoparent and nofromparent. Flow options are ignored for mainline streams.
List<String>	paths	Paths define how files are incorporated into the stream structure. Specify paths using the following format: `path_type view_path [depot_path]` where `path_type` is a single keyword, `view_path` is a file path with no leading slashes, and the optional `depot_path` is a file path beginning with `//`. The default path is `share …` Valid path types are: • `share view_path`: Specified files can be synced, submitted, and integrated to and from the parent stream. • `isolate view_path`: Specified files can be synced and submitted, but cannot be integrated to and from the parent stream. • `import view_path [depot_path]`: Specified files can be synced, but cannot be submitted or integrated to and from the parent stream. The `view_path` is mapped as in the parent stream's view, or to an (optional) `depot_path`. The `depot_path` may include a changelist specifier. That stream's client workspaces will be limited to seeing revisions at that change or lower within that depot path. For example, you can specify a depot path like this: `//depot/import/…@1000`. Revisions from changelists greater than 1000 will be automatically hidden from most commands. The changelist limits in effect for a given stream workspace are displayed in a read-only client workspace specification field called ChangeView. • `import+ view_path [depot_path]`: Functions like a standard import path, enabling you to map a path from outside the stream depot to your stream, but unlike a standard import path, you can submit changes to the files in an import+ path. • `exclude view_path`: Specified files cannot be synced, submitted or integrated to and from

Type	Name	Description
		the parent stream. By default, streams inherit their structure from the parent stream (except mainlines, which have no parent). Paths are inherited by child stream views; a child stream's path can downgrade the inherited view, but not upgrade it. (For example, a child stream can downgrade a shared path to an isolated path, but if the parent stream defines a path as isolated, its child cannot restore full access by specifying the path as shared.) Note that the `depot_path` is relevant only when the path_type is `import` or `import+`.
List<String>	remapped	Reassigns the location of workspace files. To specify the source path and its location in the workspace, use the following syntax: `view_path_1 view_path_2` where `view_path_1` and `view_path_2` are Perforce view paths (omit leading slashes and leading or embedded wildcards; terminal wildcards are fine). For example, to ensure that files are synced to the local ProjectX folder, remap as follows: … `projectX/…` Line ordering in the Remapped: field is significant: if more than one line remaps the same files, the later line takes precedence. Remappings are inherited by child streams and the workspaces associated with them.

List<String>	ignored	A list of file or directory names to be ignored in client views. For example:
		
		 ```/tmp # ignores files named "tmp"```
		 ```/tmp/... # ignores directories named "tmp"```
		 ```.tmp # ignores file names ending in .tmp```
		
		 Lines in the `Ignored:` field can appear in any order. Ignored files and directories are inherited by child stream client views.

StreamsCommand

A summary of a stream in the system, as provided by the `p4 streams` command.

Table 188. Properties

Type	Name	Description
String	stream	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form **//depotname/streamname**.
java.util.Date	update	The date the stream specification was last modified.
java.util.Date	access	The date and time that the stream specification was last accessed by any Perforce command.
String	owner	The Perforce user or group who owns the stream. The default is the user who created the stream.
String	name	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname portion of the stream path.
String	parent	The parent of this stream. Must be none if the stream's **Type:** is mainline, otherwise must be set to an existing stream identifier of the form **//depotname/streamname**.
String	type	The stream's type determines the expected flow of change. Valid stream types are **mainline**, **virtual**, **development**, and **release**. • **mainline**: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream. • **virtual**: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream. • **release**: More stable than the mainline. Release streams copy from the parent and merge to the parent. • **development**: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent.

Type	Name	Description
		• `task`: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.
String	description	Description of the stream.
String	options	Settings that configure stream behavior as follows: • `[un]locked`: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification cannot be deleted, and only its owner can modify it. The default is unlocked. • \`[all

Triggers

Defines the triggers table, like it would appear in the output to the `p4 triggers` command.

Table 189. Properties

Type	Name	Description
List<String>	triggers	A list of trigger definitions. A trigger definition contains four fields that specify the name of the trigger, the type of event that should trigger the execution of the script, the paths that should be affected by the trigger, the location of the script, and other trigger type-dependent information. When the condition specified in a trigger definition is satisfied, the associated script or program is executed. Example: `trig1 change-submit //depot/dir/…` `"/usr/bin/s1.pl %changelist%"`

Type	Name	Description
		See the Helix Versioning Engine Administrator Guide for more details on trigger definitions.

UserCommand

Create or edit Perforce user specifications and preferences.

There are three types of Perforce users: standard users, operator users, and service users. Standard users are the default, and each standard user consumes one Perforce license. The operator user type is intended for system administrators; they are subject to the same restrictions on permissions as any other user, but are further restricted in that they can run only a limited subset of Perforce commands. Service users are intended for inter-server communication in replicated and multi-server environments, and are restricted to an even smaller subset of Perforce commands. Neither operators nor service users consume Perforce licenses.

Table 190. Properties

Type	Name	Description
String	user	The Perforce username.
String	type	Type of user: standard, operator, or service. Once you set the type, you cannot change it.
String	authMethod	One of the following: perforce or ldap. Specifying perforce enables authentication using Perforce's internal db.user table or by way of an authentication trigger. This is the default unless it is overridden with the auth.default.method configurable. Specifying ldap enables authentication against AD/LDAP servers specified by the currently active LDAP configurations.
String	email	The user's email address. By default, this is user@client.
java.util.Date	update	The date and time this specification was last updated.
java.util.Date	access	The date and time this user last ran a Perforce command.
String	fullName	The user's full name.
String	jobView	Jobs matching this jobview appear on any changelists created by this user. Jobs that are

Type	Name	Description
		fixed by the changelist should be left in the changelist when it's submitted with p4 submit; other jobs should be deleted from the form before submission.
String	password	The user's password.
java.util.Date	passwordChange	The date and time of the user's last password change. If the user has no password, this field is blank.
List<String>	reviews	A list of files the user would like to review. This field can include exclusionary mappings.

UsersCommand

Table 191. Properties

Type	Name	Description
String	user	The Perforce username.
String	type	Type of user: standard, operator, or service. Once you set the type, you cannot change it.
String	email	The user's email address. By default, this is user@client.
java.util.Date	update	The date and time this specification was last updated.
java.util.Date	access	The date and time this user last ran a Perforce command.
String	fullName	The user's full name.
String	hasPassword	If 'enabled', the password has been set on the user.

JavaScript SDK Reference

Getting Started

Inside the `clients/javascript` directory of the installation is an SDK for browser or node-based JavaScript applications. The `build/helix-web-services-client.js` file is a "browserfied" distribution that exposes our core `helix_web_services_client` object. You can use this or include the project code as a node dependency, and use browserify to rebuild your own javaScript distribution. The examples below do not indicate how your code includes the client project.

You start by creating an `ApiClient` instance, with the server's URL for access. Typically, you will sign into the server as a particular user for that user's token, then continue. Most api methods are accessed via the `DefaultApi` handle from the `ApiClient.createDefaultApi` method. Models are typically accessed via the `helix_web_services_client.models` handle.

Most methods take a callback using the convention of having the first callback parameter be an error object.

```
var apiClient = new helix_web_services_client.ApiClient('https://mycompany.example.com');

var api = apiClient.createDefaultApi();

var loginRequest = new helix_web_services_client.models.LoginRequest({
    user: 'myuser',
    password: 'mypassword'
});

api.loginPost(loginRequest, function(err, loginResponse) {
    if (err) {
        console.log("server error status", err.status);
        return;
    }
    // Associate the login ticket with the apiClient.
    apiClient.apiKey = loginRepsonse.ticket;

    // At this point, our client instance is ready to make authenticated methods.
});
```

helix_web_services_client.ApiClient Reference

ApiClient properties

Table 192. Properties

Name	Type	Description
url	String	The base URL to the server
basePath	String	Path we prepend to most requests, defaults to `/api/`[VERSION].
timeout	Number	If non-zero, the time (in ms) we configure for our client connections. Defaults to 1 minute.

ApiClient constructor

Create a new ApiClient instance, configured to connect to your host.

Method Signature.

```
new ApiClient(url);
```

Table 193. Parameters

Name	Type	Description	Required
url	String	The URL of your Helix Web Services instance, e.g., https://mycompany.example.com	false (you will need to set the url property before using)

ApiClient.prototype.createDefaultApi

Creates a handle to the `DefaultApi` instance that provides most stable methods.

Method Signature.

```
DefaultApi apiClient.createDefaultApi()
```

Table 194. Returns

Type	Description
DefaultApi	See also: ["DefaultApi Reference" on page 122](#)

ApiClient.prototype.createDefaultApi

Creates a handle to the `AlphaApi` instance that provides new and unstable methods.

Method Signature.

```
DefaultApi apiClient.createAlphaApi()
```

Table 195. Returns

Type	Description
DefaultApi	See also: ["AlphaApi Reference" on page 150](#)

DefaultApi Reference

INFO: The `DefaultApi` class is not instantiated directly, but accessed via ["ApiClient.prototype.createDefaultApi" on page 122](#)

DefaultApi.prototype.configP4dsGet

The list of registered p4d servers in your cluster.

This is provided by a special set of configuration files in the system. For more information, consult the Helix Web Services user guide.

Method Signature.

```
DefaultApi.configP4dsGet(callback);
```

Table 196. Parameters

Name	Type	Description	Required
callback	function(error, Array)	The second parameter: Array of "P4dConfigId" on page 180	true

DefaultApi.prototype.loginPost

Logs into the primary authentication source.

This can either be a p4d instance or Helix Cloud, depending upon the configuration of your Helix Web Services instance.

Method Signature.

```
DefaultApi.loginPost(loginRequest, callback);
```

Table 197. Parameters

Name	Type	Description	Required
loginRequest	"LoginRequest" on	The user login and password.	true
callback	function(error, LoginResponse)	The second parameter: "LoginResponse" on page 180	true

DefaultApi.prototype.statusGet

A simple structure to monitor for "problems" an admin should take care of, and, report the current application version.

This method does not require authentication.

Method Signature.

```
DefaultApi.statusGet(callback);
```

Table 198. Parameters

Name	Type	Description	Required
callback	function(error, HWSStatus)	The second parameter: "HWSStatus" on page 176	true

DefaultApi.prototype.serverBranchesGet

Lists available branches in the system. The resources of this list are summaries of branches in the system.

Method Signature.

```
DefaultApi.serverBranchesGet(server, callback);
```

Table 199. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Array)	The second parameter: Array of "BranchesCommand" on page 154	true

DefaultApi.prototype.serverBranchesPost

Creates a new branch specification, like the `p4 branch` command.

Method Signature.

```
DefaultApi.serverBranchesPost(server, body, callback);
```

Table 200. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	"BranchCommand"	The branch specification.	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverBranchesBranchGet

Returns the branch spec details of the particular branch.

Method Signature.

```
DefaultApi.serverBranchesBranchGet(server, branch, callback);
```

Table 201. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
branch	String	The branch ID	true
callback	function(error, BranchCommand)	The second parameter: ["BranchCommand" on page 153](#)	true

DefaultApi.prototype.serverBranchesBranchPatch

Update branch specifications, similar to the `p4 branch` command. Only the specified parameters in the body will be changed.

Method Signature.

```
DefaultApi.serverBranchesBranchPatch(server, branch, body, callback);
```

Table 202. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
branch	String	The branch ID	true
body	["BranchCommand"](#)	Fields of the branch to update	true
callback	function(error, CommandResponse)	The second parameter: ["CommandResponse" on page 163](#)	true

DefaultApi.prototype.serverBranchesBranchDelete

Removes the branch specification, similar to the `p4 branch -d` command.

Method Signature.

```
DefaultApi.serverBranchesBranchDelete(server, branch, callback);
```

Table 203. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
branch	String	The branch ID	true

Name	Type	Description	Required
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverChangesGet

Lists available changes in the system. The resources of this list are summaries of changes in the system.

Method Signature.

```
DefaultApi.serverChangesGet(server, max, status, user, files, callback);
```

Table 204. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
max	Number	Limit the number of change results	false
status	String	The status of the changes, e.g., submitted	false
user	String	The user's login who submitted the change	false
files	String	Limit changes to the depot path expressions. See the changes command description.	false
callback	function(error, Array)	The second parameter: Array of "ChangesCommand" on page 156	true

DefaultApi.prototype.serverChangesChangeGet

Returns the change spec details of the particular change.

Method Signature.

```
DefaultApi.serverChangesChangeGet(server, change, callback);
```

Table 205. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
change	String	The change ID	true

Name	Type	Description	Required
callback	function(error, ChangeCommand)	The second parameter: "ChangeCommand" on page 154	true

DefaultApi.prototype.serverClientsGet

Lists available clients in the system. The resources of this list are summaries of clients in the system.

Method Signature.

```
DefaultApi.serverClientsGet(server, callback);
```

Table 206. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Array)	The second parameter: Array of "ClientsCommand" on page 161	true

DefaultApi.prototype.serverClientsPost

Creates a new client specification, like the `p4 client` command.

Method Signature.

```
DefaultApi.serverClientsPost(server, client, callback);
```

Table 207. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
client	"ClientCommand"	The client spec	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverClientsClientGet

Returns the client spec details of the particular client.

Method Signature.

```
DefaultApi.serverClientsClientGet(server, client, callback);
```

Table 208. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
client	String	The client ID	true
callback	function(error, ClientCommand)	The second parameter: "ClientCommand" on page 157	true

DefaultApi.prototype.serverClientsClientPatch

Update client specifications, similar to the `p4 client` command. Only the specified parameters in the body will be changed.

Method Signature.

```
DefaultApi.serverClientsClientPatch(server, client, body, callback);
```

Table 209. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
client	String	The client ID	true
body	"ClientCommand"	Fields of the client to update	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverClientsClientDelete

Removes the client specification, similar to the `p4 client -d` command.

Method Signature.

```
DefaultApi.serverClientsClientDelete(server, client, callback);
```

Table 210. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
client	String	The client ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverCommandsCommandGet

Execute a Perforce command that requires no input. This only allows commands that have been whitelisted on your system. See the "Configuration" on page 23 section for details.

Method Signature.

```
DefaultApi.serverCommandsCommandGet(server, command, arg, callback);
```

Table 211. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
command	String	The command name	true
arg		Command arguments	false
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverCommandsCommandPost

Execute a Perforce command that accepts input, like a spec. This only allows commands that have been whitelisted on your system. See the "Configuration" on page 23 section for details.

Method Signature.

```
DefaultApi.serverCommandsCommandPost(server, command, arg, input, callback);
```

Table 212. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
command	String	The command name	true
arg		Command arguments	false

Name	Type	Description	Required
input	"CommandRequest"	A hash used as input to the command	false
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverCountersGet

Lists available counters in the system. The resources of this list are summaries of counters in the system.

Method Signature.

```
DefaultApi.serverCountersGet(server, callback);
```

Table 213. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Array)	The second parameter: Array of "Counter" on page 164	true

DefaultApi.prototype.serverCountersCounterPut

Update counter specifications, similar to the `p4 counter` command. Only the specified parameters in the body will be changed.

Method Signature.

```
DefaultApi.serverCountersCounterPut(server, counter, body, callback);
```

Table 214. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
counter	String	The counter ID	true
body	"Counter" on page	Fields of the counter to update	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverCountersCounterGet

Returns the counter spec details of the particular counter.

Method Signature.

```
DefaultApi.serverCountersCounterGet(server, counter, callback);
```

Table 215. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
counter	String	The counter ID	true
callback	function(error, Counter)	The second parameter: "Counter" on page 164	true

DefaultApi.prototype.serverCountersCounterDelete

Removes the counter specification, similar to the `p4 counter -d` command.

Method Signature.

```
DefaultApi.serverCountersCounterDelete(server, counter, callback);
```

Table 216. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
counter	String	The counter ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverCountersCounterIncrementPost

Increments a numerical counter, similar to the `p4 counter -i` command.

Method Signature.

```
DefaultApi.serverCountersCounterIncrementPost(server, counter, callback);
```

Table 217. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
counter	String	The counter ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverDepotsGet

Lists available depots in the system. The resources of this list are summaries of depots in the system.

Method Signature.

```
DefaultApi.serverDepotsGet(server, callback);
```

Table 218. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Array)	The second parameter: Array of "DepotsCommand" on page 166	true

DefaultApi.prototype.serverDepotsPost

Creates a new depot specification, like the `p4 depot` command.

Method Signature.

```
DefaultApi.serverDepotsPost(server, depot, callback);
```

Table 219. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
depot	"DepotCommand"	The depot spec	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverDepotsDepotGet

Returns the depot spec details of the particular depot.

Method Signature.

```
DefaultApi.serverDepotsDepotGet(server, depot, callback);
```

Table 220. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
depot	String	The depot ID	true
callback	function(error, DepotCommand)	The second parameter: "DepotCommand" on page 164	true

DefaultApi.prototype.serverDepotsDepotPatch

Update depot specifications, similar to the `p4 depot` command. Only the specified parameters in the body will be changed.

Method Signature.

```
DefaultApi.serverDepotsDepotPatch(server, depot, body, callback);
```

Table 221. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
depot	String	The depot ID	true
body	"DepotCommand"	Fields of the depot to update	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverDepotsDepotDelete

Removes the depot specification, similar to the `p4 depot -d` command.

Method Signature.

```
DefaultApi.serverDepotsDepotDelete(server, depot, callback);
```

Table 222. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
depot	String	The depot ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverGroupsGet

Lists available groups in the system. The resources of this list are summaries of groups in the system.

Method Signature.

```
DefaultApi.serverGroupsGet(server, callback);
```

Table 223. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Array)	The second parameter: Array of "GroupsCommand" on page 175	true

DefaultApi.prototype.serverGroupsPost

Creates a new group specification, like the **p4 group** command.

Method Signature.

```
DefaultApi.serverGroupsPost(server, body, callback);
```

Table 224. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	"GroupCommand"	The group spec	true

Name	Type	Description	Required
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverGroupsGroupGet

Returns the group spec details of the particular group.

Method Signature.

```
DefaultApi.serverGroupsGroupGet(server, group, callback);
```

Table 225. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
group	String	The group ID	true
callback	function(error, GroupCommand)	The second parameter: "GroupCommand" on page 174	true

DefaultApi.prototype.serverGroupsGroupPatch

Update group specifications, similar to the `p4 group` command. Only the specified parameters in the body will be changed.

Method Signature.

```
DefaultApi.serverGroupsGroupPatch(server, group, body, callback);
```

Table 226. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
group	String	The group ID	true
body	"GroupCommand"	Fields of the group to update	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverGroupsGroupDelete

Removes the group specification, similar to the `p4 group -d` command.

Method Signature.

```
DefaultApi.serverGroupsGroupDelete(server, group, callback);
```

Table 227. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
group	String	The group ID	true
callback	function(error, CommandResponse)	The second parameter: ["CommandResponse" on page 163](#)	true

DefaultApi.prototype.serverJobsGet

Lists available jobs in the system. The resources of this list are summaries of jobs in the system.

Method Signature.

```
DefaultApi.serverJobsGet(server, callback);
```

Table 228. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Array)	The second parameter: Array of ["JobsCommand" on page 176](#)	true

DefaultApi.prototype.serverJobsPost

Creates a new job specification, like the `p4 job` command.

Method Signature.

```
DefaultApi.serverJobsPost(server, job, callback);
```

Table 229. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
job	"JobCommand" on	The job spec	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverJobsJobGet

Returns the job spec details of the particular job.

Method Signature.

```
DefaultApi.serverJobsJobGet(server, job, callback);
```

Table 230. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	String	The job ID	true
callback	function(error, JobCommand)	The second parameter: "JobCommand" on page 176	true

DefaultApi.prototype.serverJobsJobPatch

Update job specifications, similar to the `p4 job` command. Only the specified parameters in the body will be changed.

Method Signature.

```
DefaultApi.serverJobsJobPatch(server, job, jobCommand, callback);
```

Table 231. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	String	The job ID	true
jobCommand	"JobCommand" on	Fields of the job to update	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverJobsJobDelete

Removes the job specification, similar to the `p4 job -d` command.

Method Signature.

```
DefaultApi.serverJobsJobDelete(server, job, callback);
```

Table 232. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	String	The job ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverJobsJobFixesChangeDelete

Removes the fix record association for the job for a particular changelist.

Method Signature.

```
DefaultApi.serverJobsJobFixesChangeDelete(server, job, change, callback);
```

Table 233. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	String	The job ID	true
change	String	The change ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverJobsJobFixesChangePost

Adds a fix record to the job for a particular changelist.

Method Signature.

```
DefaultApi.serverJobsJobFixesChangePost(server, job, change, status, callback);
```

Table 234. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	String	The job ID	true
change	String	The change ID	true
status	String	Specify the job status instead of using the default. The default is typically `closed` or some other value defined in the `Presets` field specified in the `p4 jobspec` form. If the changelist to which you're linking the job been submitted, the status value is immediately reflected in the job's status. If the changelist is pending, the job status is changed on submission of the changelist, provided that the -s option is also supplied to p4 submit and the desired status appears next to the job in the p4 submit form's Jobs: field. To leave a job unchanged, use the special status of same.	false
callback	`function(error, CommandResponse)`	The second parameter: ["CommandResponse" on page 163](#)	true

DefaultApi.prototype.serverLabelsGet

Lists available labels in the system. The resources of this list are summaries of labels in the system.

Method Signature.

```
DefaultApi.serverLabelsGet(server, callback);
```

Table 235. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	`function(error, Array)`	The second parameter: Array of ["LabelsCommand" on page 177](#)	true

DefaultApi.prototype.serverLabelsPost

Creates a new label specification, like the `p4 label` command.

Method Signature.

```
DefaultApi.serverLabelsPost(server, label, callback);
```

Table 236. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
label	"LabelCommand"	The label spec	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverLabelsLabelGet

Returns the label spec details of the particular label.

Method Signature.

```
DefaultApi.serverLabelsLabelGet(server, label, callback);
```

Table 237. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
label	String	The label ID	true
callback	function(error, LabelCommand)	The second parameter: "LabelCommand" on page 177	true

DefaultApi.prototype.serverLabelsLabelPatch

Update label specifications, similar to the `p4 label` command. Only the specified parameters in the body will be changed.

Method Signature.

```
DefaultApi.serverLabelsLabelPatch(server, label, labelCommand, callback);
```

Table 238. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
label	String	The label ID	true
labelCommand	"LabelCommand" (	Fields of the label to update	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverLabelsLabelDelete

Removes the label specification, similar to the `p4 label -d` command.

Method Signature.

```
DefaultApi.serverLabelsLabelDelete(server, label, callback);
```

Table 239. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
label	String	The label ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverLoginPost

Logs into a Helix Versioning Engine (p4d) server.

Method Signature.

```
DefaultApi.serverLoginPost(server, body, callback);
```

Table 240. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	"LoginRequest" on	The user login and password.	true

Name	Type	Description	Required
callback	function(error, LoginResponse)	The second parameter: ["LoginResponse" on page 180](#)	true

DefaultApi.prototype.serverPathsGet

Lists depots, files, and directories in the system. This combines the output of the `p4 depots`, `p4 dirs`, and `p4 files` commands, depending upon your path.

Method Signature.

```
DefaultApi.serverPathsGet(server, path, callback);
```

Table 241. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
path	String	The path "under a depot" to query under, e.g., `//depot/main`. This will list the directories and files underneath that path.	false
callback	function(error, Array)	The second parameter: Array of ["Location" on page 179](#)	true

DefaultApi.prototype.serverProtectionsPut

Updates the protections table.

This method requires superuser access.

Method Signature.

```
DefaultApi.serverProtectionsPut(server, protections, callback);
```

Table 242. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
protections	["Protections" on pa](#)	The new protections table	true
callback	function(error, CommandResponse)	The second parameter: ["CommandResponse" on page 163](#)	true

DefaultApi.prototype.serverProtectionsGet

Returns a list of available protections in the system. The elements of this list are rows of the system's protections table.

This method requires superuser access.

See the output of **p4 protect** for more information.

Method Signature.

```
DefaultApi.serverProtectionsGet(server, callback);
```

Table 243. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Protections)	The second parameter: "Protections" on page 180	true

DefaultApi.prototype.serverServersGet

Lists available servers in the system. The resources of this list are summaries of servers in the system.

Method Signature.

```
DefaultApi.serverServersGet(server, callback);
```

Table 244. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Array)	The second parameter: Array of "ServersCommand" on page 182	true

DefaultApi.prototype.serverServersPost

Creates a new server specification, like the **p4 server** command.

Method Signature.

```
DefaultApi.serverServersPost(server, serverCommand, callback);
```

Table 245. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
serverCommand	*"ServerCommand"*	The server spec	true
callback	`function(error, CommandResponse)`	The second parameter: *"CommandResponse" on page 163*	true

DefaultApi.prototype.serverServersServerIdGet

Returns the server spec details of the particular server.

Method Signature.

```
DefaultApi.serverServersServerIdGet(server, serverId, callback);
```

Table 246. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
serverId	String	The server ID of the server spec	true
callback	`function(error, ServerCommand)`	The second parameter: *"ServerCommand" on page 183*	true

DefaultApi.prototype.serverServersServerIdPatch

Update server specifications, similar to the `p4 server` command. Only the specified parameters in the body will be changed.

Method Signature.

```
DefaultApi.serverServersServerIdPatch(server, serverId, serverCommand, callback);
```

Table 247. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
serverId	String	The server ID	true

Name	Type	Description	Required
serverCommand	"ServerCommand"	Fields of the server to update	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverServersServerIdDelete

Removes the server specification, similar to the `p4 server -d` command.

Method Signature.

```
DefaultApi.serverServersServerIdDelete(server, serverId, callback);
```

Table 248. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
serverId	String	The server ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverStreamsGet

Lists available streams in the system. The resources of this list are summaries of streams in the system.

Method Signature.

```
DefaultApi.serverStreamsGet(server, callback);
```

Table 249. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Array)	The second parameter: Array of "StreamsCommand" on page 190	true

DefaultApi.prototype.serverStreamsPost

Creates a new stream specification, like the `p4 stream` command.

Method Signature.

```
DefaultApi.serverStreamsPost(server, body, callback);
```

Table 250. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	"StreamCommand"	The stream spec	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverStreamsStreamGet

Returns the stream spec details of the particular stream.

Method Signature.

```
DefaultApi.serverStreamsStreamGet(server, stream, callback);
```

Table 251. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
stream	String	The stream ID	true
callback	function(error, StreamCommand)	The second parameter: "StreamCommand" on page 186	true

DefaultApi.prototype.serverStreamsStreamPatch

Update stream specifications, similar to the `p4 stream` command. Only the specified parameters in the body will be changed.

Method Signature.

```
DefaultApi.serverStreamsStreamPatch(server, stream, body, callback);
```

Table 252. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
stream	String	The stream ID	true
body	"StreamCommand"	Fields of the stream to update	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverStreamsStreamDelete

Removes the stream specification, similar to the `p4 stream -d` command.

Method Signature.

```
DefaultApi.serverStreamsStreamDelete(server, stream, callback);
```

Table 253. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
stream	String	The stream ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverTriggersPut

Updates the triggers table.

This method requires superuser access.

Method Signature.

```
DefaultApi.serverTriggersPut(server, triggers, callback);
```

Table 254. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
triggers	"Triggers" on page	The new triggers table	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverTriggersGet

Returns a list of available triggers in the system. The elements of this list are rows of the system's triggers table.

This method requires superuser access.

Method Signature.

```
DefaultApi.serverTriggersGet(server, callback);
```

Table 255. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Triggers)	The second parameter: "Triggers" on page 191	true

DefaultApi.prototype.serverUsersGet

Lists available users in the system. The resources of this list are summaries of users in the system.

Method Signature.

```
DefaultApi.serverUsersGet(server, includeService, max, callback);
```

Table 256. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
includeService		If true, shows service users in the list.	false
max	Number	Cap the number of users reported to this amount.	false
callback	function(error, Array)	The second parameter: Array of "UsersCommand" on page 193	true

DefaultApi.prototype.serverUsersPost

Creates a new user specification, like the `p4 user` command.

Method Signature.

```
DefaultApi.serverUsersPost(server, body, callback);
```

Table 257. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	"UserCommand" o	The user spec	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverUsersUserGet

Returns the user spec details of the particular user.

Method Signature.

```
DefaultApi.serverUsersUserGet(server, user, callback);
```

Table 258. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
user	String	The user ID	true
callback	function(error, UserCommand)	The second parameter: "UserCommand" on page 192	true

DefaultApi.prototype.serverUsersUserPatch

Update user specifications, similar to the `p4 user` command. Only the specified parameters in the body will be changed.

Method Signature.

```
DefaultApi.serverUsersUserPatch(server, user, body, callback);
```

Table 259. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
user	String	The user ID	true
body	"UserCommand" o	Fields of the user to update	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

DefaultApi.prototype.serverUsersUserDelete

Removes the user specification, similar to the `p4 user -d` command.

Method Signature.

```
DefaultApi.serverUsersUserDelete(server, user, callback);
```

Table 260. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
user	String	The user ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

AlphaApi Reference

INFO: The `AlphaApi` class is not instantiated directly, but accessed via "ApiClient.prototype.createDefaultApi" on page 122

AlphaApi.prototype.serverChangesPost

Create a new changelist that can affect multiple files using different kinds of actions. If you require the ability to integrate or move, for example, you can use this method.

Method Signature.

```
AlphaApi.serverChangesPost(server, changelistRequest, callback);
```

Table 261. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
changelistRequest	"ChangelistRequest	Description of changes to make	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

AlphaApi.prototype.serverGitFusionReposGet

Lists all configured repositories readable by the current user. .Method Signature

```
AlphaApi.serverGitFusionReposGet(server, callback);
```

Table 262. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
callback	function(error, Array)	The second parameter: Array of "GitFusionRepoId" on page 169	true

AlphaApi.prototype.serverGitFusionReposPost

Submits a [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file to create or update a repository configuration.

If the repository does not exist or has been previously deleted, this method saves contents of the config file to a new **p4gf_config** file. If the repository has already been initialised, this method replaces all of the file contents of the specified repository's **p4gf_config** file.

Method Signature.

```
AlphaApi.serverGitFusionReposPost(server, body, callback);
```

Table 263. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	"GitFusionRepoCo	The new configuration	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

AlphaApi.prototype.serverGitFusionReposRepoGet

Return configuration for the specified repository. Grabs and returns contents of the [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file for given repository.

Method Signature.

```
AlphaApi.serverGitFusionReposRepoGet(server, repo, callback);
```

Table 264. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
repo	String	The Git Fusion Repo ID	true
callback	function(error, GitFusionRepoConf	The second parameter: "GitFusionRepoConfig" on page 169	true

AlphaApi.prototype.serverGitFusionReposRepoPatch

Updates values in the repository configuration. This method will find all specified parameters and update each value for the specified repository's configuration file.

Method Signature.

```
AlphaApi.serverGitFusionReposRepoPatch(server, repo, body, callback);
```

Table 265. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
repo	String	The Git Fusion Repo ID	true
body	"GitFusionRepoCor	The new configuration	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

AlphaApi.prototype.serverGitFusionReposRepoDelete

Deletes the repository configuration (The [p4gf_config file](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j)). Contents of the repository are not deleted from Perforce.

Method Signature.

```
AlphaApi.serverGitFusionReposRepoDelete(server, repo, callback);
```

Table 266. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
repo	String	The Git Fusion Repo ID	true
callback	function(error, CommandResponse)	The second parameter: "CommandResponse" on page 163	true

helix_web_services_client.models Reference

BranchCommand

Models the output of a `p4 branch` command.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_branch.html).

Table 267. Properties

Type	Name	Description
String	branch	The branch name, as provided on the command line.
String	owner	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.
Date	access	The date the branch mapping was last accessed.
Date	update	The date the branch mapping was last changed.
String	options	Either `unlocked` (the default) or `locked`. If `locked`, only the `Owner:` can modify the branch mapping, and the mapping can't be deleted until it is `unlocked`.
String	description	A short description of the branch's purpose.
Array of String	view	A set of mappings from one set of files in the depot (the source files) to another set of files in the depot (the target files). The view maps from one location in the depot to another; it can't refer to a client workspace. `For example, the branch view `//depot/main/... //depot/r2.1/...` maps all the files under `//depot/main` to `//depot/r2.1`.`

BranchesCommand

A reference to a branch mapping known to the system.

Table 268. Properties

Type	Name	Description
String	branch	The branch name, as provided on the command line.
String	owner	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.
Date	access	The date the branch mapping was last accessed.
Date	update	The date the branch mapping was last changed.
String	options	Either `unlocked` (the default) or `locked`. If `locked`, only the `Owner:` can modify the branch mapping, and the mapping can't be deleted until it is `unlocked`.
String	description	A short description of the branch's purpose.

ChangeCommand

A changelist specification.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_change.html).

Table 269. Properties

Type	Name	Description
String	change	Contains the changelist number if editing an existing changelist, or `new` if creating a new changelist.
String	client	Name of current client workspace
Date	date	Date the changelist was last modified.
String	user	Name of the change owner. The owner of an empty pending changelist (that is, a pending changelist without any files in it) can transfer ownership of the changelist to another existing user either by editing this field, or by using the -U user option. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does

Type	Name	Description
		not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
String	status	`pending`, `shelved`, `submitted`, or `new`. Not editable by the user.
		The status is `new` when the changelist is created, `pending` when it has been created but has not yet been submitted to the depot, `shelved` when its contents are shelved, and `submitted` when its contents have been stored in the depot.
String	description	Textual description of changelist.
		If you do not have access to a restricted changelist, the description is replaced with a "no permission" message.
Array of String	jobs	A list of jobs that are fixed by this changelist.
String	type	Type of change: `restricted` or `public`.
		The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change.
		Public changes are displayed without restrictions.
		By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.
Array of String	files	The list of files submitted in this changelist.
String	importedBy	Displays the name of the user who ran the p4 fetch, p4 push, or p4 unzip command that imported this change into the server.
		This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied.
		In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.
String	identify	Contains a label which uniquely identifies this changelist across all servers where it has been fetched, pushed, or unzipped.

Type	Name	Description
		This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied.
		In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.

ChangesCommand

Table 270. Properties

Type	Name	Description
String	change	The changelist ID
Date	date	Last modification time of the changelist
String	user	The owner of the changelist
String	client	Name of current client workspace.
String	status	`pending`, `shelved`, `submitted`, or `new`. Not editable by the user. The status is `new` when the changelist is created, `pending` when it has been created but has not yet been submitted to the depot, `shelved` when its contents are shelved, and `submitted` when its contents have been stored in the depot.
String	type	Type of change: `restricted` or `public`. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change. Public changes are displayed without restrictions. By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.
String	path	Depot paths affected by this changelist
String	description	Textual description of changelist.

Type	Name	Description
		If you do not have access to a restricted changelist, the description is replaced with a "no permission" message.

ChangelistRequest

Table 271. Properties

Type	Name	Description
String	description	
String	stream	Optional stream ID to use in case you want to edit files in a stream.
Array of "ChangelistAction"	actions	

ChangelistAction

Table 272. Properties

Type	Name	Description
String	depotFile	The target file path to edit.
String	fromDepotFile	For "branch" or "move" actions, this indicates the source file location.
String	actionType	One of "upload", "branch", "move", or "delete"
String	content	Base64-encoded content
Number	requireVersion	If set, we will only operate if this is the current version of the file.

ClientCommand

The client workspace specification and its view.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html).

Table 273. Properties

Type	Name	Description
String	client	The client workspace name, as specified in the P4CLIENT environment variable or its equivalents.

Type	Name	Description
String	owner	The name of the user who owns the workspace. The default is the user who created the workspace.
		The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
Date	update	The time the workspace specification was last modified.
Date	access	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)
String	host	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host.
		The hostname must be provided exactly as it appears in the output of p4 info when run from that host.
		This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.
String	description	A textual description of the workspace. The default text is Created by owner.
String	root	The directory (on the local host) relative to which all the files in the `View:` are specified. The default is the current working directory. The path must be specified in local file system syntax.
		If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to map files to multiple drives.
Array of String	altRoots	Up to two optional alternate client workspace roots.
		Perforce applications use the first of the main and alternate roots that match the application's current working directory. Use the p4 info command to display the root being used.

Type	Name	Description
		This enables users to use the same Perforce client workspace specification on multiple platforms, even those with different directory naming conventions.
		If you are using multiple or alternate workspace roots (the AltRoots: field), you can always tell which root is in effect by looking at the Client root: reported by p4 info.
		If you are using a Windows directory in any of your workspace roots, you must specify the Windows directory as your main workspace root and specify your other workspace roots in the AltRoots: field.
String	options	A set of seven switches that control particular workspace options.
		See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
String	submitOptions	Options to govern the default behavior of p4 submit.
		• submitunchanged
		All open files (with or without changes) are submitted to the depot. This is the default behavior of Perforce.
		• submitunchanged+reopen
		All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist.
		• revertunchanged
		Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted.
		• revertunchanged+reopen
		Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist.
		• leaveunchanged
		Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist.

Type	Name	Description
		• leaveunchanged+reopen Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged+reopen, except that no unchanged files are submitted to the depot.
String	lineEnd	Configure carriage-return/linefeed (CR/LF) conversion. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
String	stream	Associates the workspace with the specified stream. Perforce generates the view for stream-associated workspaces: you cannot modify it manually.
String	streamAtChange	A changelist number that sets a back-in-time view of a stream. When StreamAtChange is set, running p4 sync (when called with no arguments) updates the workspace to files at this changelist revision, instead of the head revision. You cannot submit changes (p4 submit returns an error) when StreamAtChange is set, because the workspace view no longer reflects the current stream inheritance. This field is ignored unless the Stream field is also set to a valid stream.
String	serverID	If set, restricts usage of the workspace to the named server. If unset, use is allowed on master server and on any replicas of the master other than Edge servers.
Array of String	view	Specifies the mappings between files in the depot and files in the workspace. A new view takes effect on the next p4 sync operation.
Array of String	changeView	Restricts access to depot paths to a particular point in time. Files specified for the ChangeView field are read-only: they may be opened but not submitted. For example: `//depot/path/…@1000` Revisions of the files in the specified path will not be visible if they were submitted after the specified changelist

Type	Name	Description
		number. Files matching a ChangeView path may not be submitted.
String	type	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts.
		Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.

ClientsCommand

Table 274. Properties

Type	Name	Description
String	client	The client workspace name, as specified in the P4CLIENT environment variable or its equivalents.
String	owner	The name of the user who owns the workspace. The default is the user who created the workspace.
		The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
Date	update	The time the workspace specification was last modified.
Date	access	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)
String	host	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host.
		The hostname must be provided exactly as it appears in the output of p4 info when run from that host.
		This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the

Type	Name	Description
		host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.
String	description	A textual description of the workspace. The default text is Created by owner.
String	root	The directory (on the local host) relative to which all the files in the `View:` are specified. The default is the current working directory. The path must be specified in local file system syntax. If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to map files to multiple drives. additionalProperties:
String	type	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts. Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.
String	options	A set of seven switches that control particular workspace options. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
String	submitOptions	Options to govern the default behavior of p4 submit. • submitunchanged All open files (with or without changes) are submitted to the depot. This is the default behavior of Perforce. • submitunchanged+reopen

Type	Name	Description
		All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist. • revertunchanged Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted. • revertunchanged+reopen Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist. • leaveunchanged Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist. • leaveunchanged+reopen Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged+reopen, except that no unchanged files are submitted to the depot.
String	lineEnd	Configure carriage-return/linefeed (CR/LF) conversion. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
String	stream	Associates the workspace with the specified stream. Perforce generates the view for stream-associated workspaces: you cannot modify it manually.

CommandResponse

A generic container for responses from the p4d server that we have yet to completely classify.

Table 275. Properties

Type	Name	Description
Array of object	results	A collection of maps that have various values set by p4d.

CommandRequest

A single map typically defines input to generic command methods.

Table 276. Properties

Type	Name	Description
Object	object	Don't use this. It's a kludge around a bug in the Java client code generator

Counter

A persistent variable in the server.

Table 277. Properties

Type	Name	Description
String	counter	The variable name
String	value	The variable value. Many variables are numerical in nature, which allow you to do atomic increment operations in method calls instead of having to fetch, increment, and save.

DepotCommand

The depot specification, which is the shared repository Perforce stores files in.

Table 278. Properties

Type	Name	Description
String	depot	The depot name.
String	owner	The user who owns the depot. By default, this is the user who created the depot. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
String	description	A short description of the depot's purpose. Optional.

Type	Name	Description
String	type	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`.
		A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server.
		A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy.
		A remote depot references files that reside on other servers, and cannot be written to.
		The spec depot, if present, automatically archives edited forms.
		The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.
		An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries.
		A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.
String	address	If the `Type:` is `remote`, the address should be the P4PORT address of the remote server. If the `Type:` is local or spec, this field is ignored.
String	suffix	If the Type: is spec, this field holds an optional suffix for generated paths to objects in the spec depot.
		The default suffix is .p4s. You do not need a suffix to use the spec depot, but supplying a file extension to your Perforce server's versioned specs enables users of GUI client software to associate Perforce specifications with a preferred text editor. If the Type: is local or remote, this field is ignored.
String	streamDepth	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following the depot name of a stream.]
		This field is used when streams are being created. The default is 1, matching the traditional stream name. You

Type	Name	Description
		cannot update this value once streams or archive data exist in a depot.
String	map	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the ... wildcard; for example, a local depot new might have a Map: of `new/...`. If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, `//depot/new/rel2/...`. This directory will be the root of the local representation of the remote depot.
Array of String	specMap	For spec depots, an optional description of which specs should be saved, expressed as a view.

DepotsCommand

A summary of depots in the system, with information provided by the `p4 depots` command.

Table 279. Properties

Type	Name	Description
String	depot	The depot name.
String	map	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the ... wildcard; for example, a local depot new might have a Map: of `new/...`. If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, `//depot/new/rel2/...`. This directory will be the root of the local representation of the remote depot.
String	type	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`. A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server. A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy. A remote depot references files that reside on other servers, and cannot be written to. The spec depot, if present, automatically archives edited forms.

Type	Name	Description
		The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.
		An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries.
		A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.
String	streamDepth	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following the depot name of a stream.]
		This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.
String	description	A short description of the depot's purpose. Optional.

DirsCommand

Table 280. Properties

Type	Name	Description
String	dir	

FilesCommand

Table 281. Properties

Type	Name	Description
String	depotFile	
String	revision	
String	change	
String	action	
Date	time	

Type	Name	Description
String	type	

FstatCommand

Detailed information about each file, as provided by the `p4 fstat` command.

Table 282. Properties

Type	Name	Description
String	depotFile	Depot path to file. For files containing special characters, the filename is displayed containing the ASCII expression of the character's hexadecimal value.
String	movedFile	Name in depot of moved to/from file.
String	shelved	Set to **shelved** if file is shelved.
String	headAction	Action taken at head revision, if in depot. One of: add, edit, delete, branch, move/add, move/delete, integrate, import, purge, or archive.
String	headChange	Head revision changelist number, if in depot.
String	headRev	Head revision number, if in depot.
String	headType	Head revision type, if in depot.
String	headCharset	Head charset, for unicode files.
Date	headTime	Head revision changelist time, if in depot. Time is measured in seconds since 00:00:00 UTC, January 1, 1970.
Date	headModTime	Head revision modification time (the time that the file was last modified on the client before submit), if in depot.
String	movedRev	Head revision of moved file.
String	digest	MD5 digest of a file.
String	fileSize	File length in bytes.
String	actionOwner	User who opened the file, if open.
String	resolved	The number, if any, of resovled integration records.
String	unresolved	The number, if any, of unresolved integration records.
String	reresolvable	The number, if any, of re-resolvable integration records.

Type	Name	Description
Array of String	otherOpens	For each user with the file open, the workspace and user with the open file.
Array of String	otherLocks	For each user with the file locked, the workspace and user holding the lock.
Array of String	otherActions	For each user with the file open, the action taken.
Array of String	otherChanges	The changelist number with this file open.
Array of String	resolveActions	Pending integration action.
Array of String	resolveBaseFiles	Pending base files.
Array of String	resolveBaseRevs	Pending base revision numbers.
Array of String	resolveFromFiles	Pending from files.
Array of String	resolveStartFromRe	Pending starting revisions.
Array of String	resolveEndFromRe	Pending ending revisions.

GitFusionRepoId

Table 283. Properties

Type	Name	Description
String	id	An identifier for the repository that can be used safely within URL paths.
String	name	The repository name, which can be path-like.

GitFusionRepoConfig

Table 284. Properties

Type	Name	Description
String	name	The repository name, which can be path-like.
String	description	Repo description returned by the @list command.
"GitFusionRepoGlo	globalOverrides	
Array of "GitFusionRepoBra	branches	

GitFusionRepoBranchConfig

Defines a unique Git Fusion branch.

Table 285. Properties

Type	Name	Description
String	gitBranchId	Alphanumeric ID for the git branch. *Do not change this value once this repo has been cloned*.
String	gitBranchName	Defines a name specified in a local repo for a Git branch. A valid Git branch name. Do not edit this value after you clone the repo.
Array of String	view	Defines a Perforce workspace view mapping that maps Perforce depot paths (left side) to Git work tree paths (right side). Correctly formed mapping syntax; must not include any Perforce stream or spec depots, and all depot paths on the right side must match exactly across all branch definitions. You can add and remove only certain types of Perforce branches from this view after you clone the repo.
String	stream	Defines a Perforce stream that maps to the Git branch. Provide a stream name using the syntax //streamdepot/mystream. A Git Fusion branch can be defined as a view or a stream but not both. If your branch is defined as stream, it can include only one stream.
String	readOnly	Prohibit git pushes that introduce commits to the branch.

GitFusionRepoGlobalOverrides

A list of per-repo settings that override global settings.

Table 286. Properties

Type	Name	Description
String	charset	Defines the default Unicode setting that Git Fusion applies to new repos. This setting is valid only when Git Fusion interacts with a Unicode-enabled Perforce server. (Defaults to `UTF-8`).
String	depotPathRepoCrea	Allow Git users to create new repos by pushing/pulling a git url which specifies a Perforce depot path. This is similar to creating a repo from a p4 client.

Type	Name	Description
String	depotPathRepoCre:	Restrict which authenticated Git pushers are allowed to create new repos when depot-path-repo-creation-enable is enabled.
String	changeOwner	Defines whether Git Fusion assigns either the Git commit author or the Git pusher as the owner of a pushed change (submit).
String	enableGitBranchCr	Defines whether Git Fusion creates a new branch of Perforce depot file hierarchy for each copied branch of Git workspace history, including Git task branches as Git Fusion anonymous branches.
String	enableSwarmRevie	Permits branch creation for Swarm reviews, even when enable-git-branch-creation is disabled.
String	enableGitMergeCo	Defines whether Git Fusion copies merge commits and displays them in Perforce as integrations between Perforce branches.
String	enableGitSubmodu	Defines whether Git Fusion allows Git submodules to be pushed to Perforce.
String	ignoreAuthorPermi	Defines whether Git Fusion evaluates both the author's and pusher's Perforce write permissions during a push or evaluates only the pusher's permissions.
String	preflightCommit	Enables you to trigger pre-flight commit scripts that enforce local policy for Git pushes. This can be especially useful if you have Perforce submit triggers that could reject a push and damage the repository.
String	readPermissionChe	Enables you to require that Git clone, pull, or fetch requests check the Perforce protections table for the puller's read permission on the files being pulled.
String	gitMergeAvoidance	If the Perforce service includes any changelists submitted by Git Fusion 13.2 or earlier, you can prevent unnecessary merge commits by setting this key to the number of the last changelist submitted before your site upgraded to a later version of Git Fusion.
String	jobLookup	Set the format for entering Perforce jobs in Git commit descriptions so that they are recognized by Git Fusion and appear in Perforce changelists as fixes. By default, job IDs whose string starts with "job" (as in job123456) are passed through to the changelist description and job field. Use this option if you want Git Fusion to recognize additional expressions, such as JIRA issue IDs.

Type	Name	Description
String	depotBranchCreatic	Allow Git users to create new fully-populated depot branches within Perforce.
String	depotBranchCreatic	Restrict the authenticated Git pushers who are allowed to create new fully-populated depot branches, if depotBranchCreationEnable is enabled.
String	depotBranchCreatic	Tell Git Fusion where to create new fully-populated depot branches, if depotBranchCreationEnable is enabled. Default path is `//depot/[repo]/[git_branch_name]`.
String	depotBranchCreatic	Set how the depot path set in depotBranchCreationDepotPath should appear in Git. Enter a Perforce view specification that maps Perforce depot paths (left side) to Git work tree paths (right side). Perforce depot paths are relative to the root set in depotBranchCreationDepotPath. The default … … maps every file under the depotBranchCreationDepotPath root to Git. Right side paths must match the right side for every other branch already defined within a repo.
String	enableGitFindCopic	When Git reports a copy file action, store that action in Perforce as a p4 integ. Often set in tandem with enableGitFindRenames. No/Off/0%: Do not use Git's copy detection. Treat all possible file copy actions as p4 add actions. 1%-100%: Use Git's copy detection. Value passed to git diff-tree --find-copies=n. Git Fusion also adds --find-copies-harder whenever adding --find-copies.
String	enableGitFindRena:	When Git reports a rename (also called move) file action, store that in Perforce as a p4 move. Often set in tandem with enableGitFindCopies. `No`/`Off`/`0%`: Do not use Git's rename detection. Treat all possible file rename actions as independent p4 delete and p4 add actions. `1%`-`100%`: Use Git's rename detection. Value passed to git diff-tree --find-renames=n.

Type	Name	Description
String	enableStreamImpor	Enables you to convert Perforce stream import paths to Git submodules when you clone a Git Fusion repository. If set to Yes, you must also set either httpUrl or sshUrl.
String	httpUrl	The URL used by Git to clone a repository from Git Fusion over HTTP. This property is required if you want to use Perforce stream import paths as git submodules and you use HTTP(S).
String	sshUrl	The "URL" used by Git to clone a repository from Git Fusion using SSH. This property is required if you want to use Perforce stream import paths as git submodules and you use SSH.
String	emailCaseSensitivit	Defines whether Git Fusion pays attention to case when matching Git user email addresses to Perforce user account email addresses during the authorization check.
String	authorSource	Defines the source that Git Fusion uses to identify the Perforce user associated with a Git push. Defaults to `git-email`. Use any one of the following values: • `git-email`: Use the email address of the Git author to look for a Perforce user account with the same email address. Git Fusion consults the p4gf_usermap file first, and if that fails to produce a match, it scans the Perforce user table. • `git-user`: Use the user.name field in the Git commit. This is the part of the author field before the email address. • `git-email-account`: Use the account portion of the Git author's email address. If the Git author's email value is `samwise@the_shire.com`, Git Fusion uses the Perforce account samwise. You can also tell Git Fusion to iterate through multiple source types until it finds a matching Perforce account. Specify the source types in order of precedence, separated by commas. For example: git-user, git-email-account, git-email.
String	limitSpaceMb	Natural number representing the number of megabytes of disk space that can be consumed by any single repo. This value does not include the spaced consumed on the Perforce server.

Type	Name	Description
String	limitCommitsRecei	Natural number representing the maximum number of commits allowed in a single push.
String	limitFilesReceived	Natural number representing the maximum number of files allowed in a single push.
String	limitMegabytesRec	Natural number representing the maximum number of megabytes allowed in a single push.

GroupCommand

Add or delete users from a group, or set the maxresults, maxscanrows, maxlocktime, and timeout limits for the members of a group.

Table 287. Properties

Type	Name	Description
String	group	The name of the group, as entered on the command line.
String	maxResults	The maximum number of results that members of this group can access from the service from a single command. The default value is **unset**.
String	maxScanRows	The maximum number of rows that members of this group can scan from the service from a single command. The default value is **unset**.
String	maxLockTime	The maximum length of time (in milliseconds) that any one operation can lock any database table when scanning data. The default value is **unset**.
String	maxOpenFiles	The maximum number of files that a member of a group can open using a single command.
String	timeout	The duration (in seconds) of the validity of a session ticket created by p4 login. The default value is 43,200 seconds (12 hours). To create a ticket that does not expire, set the Timeout: field to **unlimited**.
String	passwordTimeout	The length of time (in seconds) for which passwords for users in this group remain valid. To disable password aging, use a value of unset.
String	ldapConfig	The LDAP configuration to use when populating the group's user list from an LDAP query.
String	ldapSearchQuery	The LDAP query used to identify the members of the group.

Type	Name	Description
String	ldapUserAttribute	The LDAP attribute that represents the user's username.
Array of String	subgroups	Names of other Perforce groups. To add all users in a previously defined group to the group you're presently working with, include the group name in the Subgroups: field of the p4 group form. Note that user and group names occupy separate namespaces, and thus, groups and users can have the same names. Every member of any previously defined group you list in the Subgroups: field will be a member of the group you're now defining.

Array of String	owners	Names of other Perforce users.
		
		`Group owners without super access are permitted to administer this group, provided that they use the -a option.`
		
		`Group owners are not necessarily members of a group; if a group owner is to be a member of the group, the userid must also be added to the Users: field.`
		
		`The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.`

Array of String	users	The Perforce usernames of the group members.

GroupsCommand

A list of entries that can show the layout how users are associated with the different groups in the system.

Table 288. Properties

Type	Name	Description
String	user	
String	group	
String	isSubGroup	
String	isOwner	
String	isUser	
String	maxResults	

Type	Name	Description
String	maxScanRows	
String	maxLockTime	
String	maxOpenFiles	
String	timeout	
String	passTimeout	

HWSStatus

Table 289. Properties

Type	Name	Description
String	status	When "OK" the server should be considered to be operating normally
String	version	The version of Helix Web Services server.

JobCommand

A defect, enhancement request, or other job specification.

The actual fields in a job can be edited by a superuser in your system. The default set of fields in a system are Job, Status, User, Date, and Description.

Table 290. Properties

Type	Name	Description
String	Job	The job name.

JobsCommand

A summary of jobs in the system.

The actual fields in a job can be edited by a superuser in your system. The default set of fields in a system are Job, Status, User, Date, and Description. Fields in the output of this command may be missing if the superuser removed User, Status, Date, or Description.

Table 291. Properties

Type	Name	Description
String	Job	The job name.

LabelsCommand

Table 292. Properties

Type	Name	Description
String	label	The label name.
Date	update	The date the label specification was last modified.
Date	access	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)
String	owner	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label.
		The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
String	options	Options to control behavior and storage location of labels.
		• locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync.
		• autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels.
String	description	An optional description of the label's purpose.

LabelCommand

A label specification.

Labels can be either automatic or static. Automatic labels refer to the revisions provided in the View: and Revision: fields. Static labels refer only to those specific revisions tagged by the label by means of either the p4 labelsync or p4 tag commands.

Table 293. Properties

Type	Name	Description
String	label	The label name.
String	owner	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
Date	update	The date the label specification was last modified.
Date	access	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)
String	description	An optional description of the label's purpose.
String	options	Options to control behavior and storage location of labels. • locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync. • autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels.
String	revision	An optional revision specification for an automatic label. If you use the # character to specify a revision number, you must use quotes around it in order to ensure that the # is parsed as a revision specifier, and not as a comment field in the form.
Array of String	view	A list of depot files that can be tagged with this label. No files are actually tagged until **p4 labelsync** is invoked. Unlike client views or branch views, which map one set of files to another, label views consist of a simple list of depot files.

Type	Name	Description
String	serverID	If set, restricts usage of the label to the named server. If unset, this label may be used on any server.

Location

A consolidated mechanism for identifying something that generally has a path in the system.

Each location references either a depot, a dir, or a file.

Table 294. Properties

Type	Name	Description
String	depotPath	An absolute depot path specification.
"DepotsCommand"	depot	
"DirsCommand" or	dir	
"FilesCommand" o	file	
"FstatCommand" o	fstat	
String	content	If this location indicates a single file, this can be set with the Base64-encoded content of the file.

LoginRequest

Captures the login information we need for logging into either a p4d server or our "authentication source".

Table 295. Properties

Type	Name	Description
String	user	Usually the Perforce username
String	password	
Array of "ServerLoginReque	serverLogins	

ServerLoginRequest

Table 296. Properties

Type	Name	Description
String	id	The server's ID

Type	Name	Description
String	user	
String	password	

LoginResponse

Either of our login methods return a ticket, which is then used as a password in a basic authentication scheme.

When this is returned from the explicit p4d login, this is a host unlocked ticket, acceptable for using with a local client.

Table 297. Properties

Type	Name	Description
String	ticket	

P4dConfigId

Identification of servers the Helix Web Services instance can connect to.

Table 298. Properties

Type	Name	Description
String	id	A simple string identifier (alphanumeric characters only, please)
String	name	A display string, not guaranteed to be unique
String	description	A simple textual description, for potential selection by clients.

Protections

Displays the information stored in the `p4 protect` command.

Table 299. Properties

Type	Name	Description
Array of String	protections	Each item in the protections array is a line in the protections table, and is split into five columns.
		1. Access level or mode. One of the access levels list, read, open, write, admin, super, review; or one of the rights =read, =open, =write, and =branch,

Type	Name	Description
		2. Either **user** or **group**, to indicate what's identified by this entry.
		3. The group name or user name. To grant permission to all users, use a wildcard with just an asterix symbol.
		4. The IP address of the client host.
		5. The depot file path, which can contain wildcards. To exclude this mapping from the permission set, use a dash - as the first character of this value.
		IPv6 addresses and IPv4 addresses are also supported. You can use the * wildcard to refer to all IP addresses, but only when you are not using CIDR notation.
		If you use the * wildcard with an IPv6 address, you must enclose the entire IPv6 address in square brackets. For example, [2001:db8:1:2:*] is equivalent to [2001:db8:1:2::]/64. Best practice is to use CIDR notation, surround IPv6 addresses with brackets, and to avoid the * wildcard.
		How the system forms host addresses depends on the setting of the dm.proxy.protects variable. By default, this variable is set to 1. This means that if the client host uses some intermediary (proxy, broker, replica) to access the server, the proxy- prefix is prepended to the client host address to indicate that the connection is not direct. If you specify proxy-* for the Host field, that will affect all connections made via proxies, brokers, and replicas. A value like proxy-10.0.0.5 identifies a client machine with an IP address of 10.0.0.5 that is connected to the server through an intermediary.
		Setting the dm.proxy.protects variable to 0, removes the proxy- prefix and allows you to write a single set of protection entries that apply both to directly-connected clients as well as to those that connect via an intermediary. This is more convenient but less secure if it matters that a connection is made using an intermediary. If you use this setting, all intermediaries must be at release 2012.1 or higher.

ServersCommand

Table 300. Properties

Type	Name	Description
String	serverID	A unique identifier for this server. This must match the contents of the server's `server.id` file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.
String	type	Server executable type. One of the following: `server`, `proxy`, `broker`, `identifier`, `admin`. Each type may offer one or more services, defined in the `services` property.
String	services	The `server` type server provides the following services: • standard - a standard Perforce server • replica - a read-only replica server • commit-server - central server in distributed installation • edge-server - node in distributed installation • forwarding-replica - a replica configured to forward commands that involve database writes to a master server • build-server - a replica that supports build automation and build farm integration • P4AUTH - a server that provides authentication • P4CHANGE - a server that provides change numbering • depot-master - commit-server with automated failover • depot-standby - standby replica of the depot-master • workspace-server - node in a cluster installation • standby - read-only replica server that uses p4 journalcopy • forwarding-standby - forwarding replica server that uses p4 journalcopy The `proxy` type server provides a p4p caching proxy.

Type	Name	Description
		The **broker** type server provides the following services: • broker - a p4broker process • workspace-router - routing broker for a cluster The services field for the **identifier** type server specifies the existence of the cluster, and has the value **cluster**. The name of the cluster is then drawn from the ServerID field. The **admin** type server provides the following services: • hxca-server - the admin server for a Helix cluster. • zookeeper-server - ZooKeeper server for a cluster
String	name	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.
String	address	The P4PORT used by this server.
String	description	An optional description for this server.
String	user	The service user name used by the server.

ServerCommand

The Perforce server specification describes the high-level configuration and intended usage of a Perforce server. For installations with only one Perforce server, the server specification is optional.

Table 301. Properties

Type	Name	Description
String	serverID	A unique identifier for this server. This must match the contents of the server's **server.id** file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.
String	type	Server executable type. One of the following: **server**, **proxy**, **broker**, **identifier**, **admin**. Each type may offer one or more services, defined in the **services** property.
String	services	The **server** type server provides the following services: • standard - a standard Perforce server

Type	Name	Description
		• replica - a read-only replica server
		• commit-server - central server in distributed installation
		• edge-server - node in distributed installation
		• forwarding-replica - a replica configured to forward commands that involve database writes to a master server
		• build-server - a replica that supports build automation and build farm integration
		• P4AUTH - a server that provides authentication
		• P4CHANGE - a server that provides change numbering
		• depot-master - commit-server with automated failover
		• depot-standby - standby replica of the depot-master
		• workspace-server - node in a cluster installation
		• standby - read-only replica server that uses p4 journalcopy
		• forwarding-standby - forwarding replica server that uses p4 journalcopy
		The `proxy` type server provides a p4p caching proxy.
		The `broker` type server provides the following services:
		• broker - a p4broker process
		• workspace-router - routing broker for a cluster
		The services field for the `identifier` type server specifies the existence of the cluster, and has the value `cluster`. The name of the cluster is then drawn from the ServerID field.
		The `admin` type server provides the following services:
		• hxca-server - the admin server for a Helix cluster.
		• zookeeper-server - ZooKeeper server for a cluster
String	name	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.

Type	Name	Description
String	address	The P4PORT used by this server.
String	externalAddress	For an edge server, this optional field specifies the external address used for connections to a commit server. This field must be set for the edge server to enable parallel submits in a federated environment.
String	description	An optional description for this server.
String	user	The service user name used by the server.
String	clientDataFilter	For a replica server, this optional field can contain one or more patterns describing how active client workspace metadata is to be filtered. Active client workspace data includes have lists, working records, and pending resolves. To include client data, use the syntax: **//client-pattern/…** To exclude client data, use the syntax: **-//client-pattern/ …** All patterns are specified in client syntax.
String	revisionDataFilter	For a replica server, this optional field can contain one or more patterns describing how submitted revision metadata is to be filtered. Submitted revision data includes revision records, integration records, label contents, and the files listed in submitted changelists. To include depot data, use the syntax: To exclude depot data, use the syntax: -//depot/pattern/ … All patterns are specified in depot syntax.
String	archiveDataFilter	For a replica server, this optional field can contain one or more patterns describing the policy for automatically scheduling the replication of file content. If this field is present, only those files described by the pattern are automatically transferred to the replica; other files are not transferred until they are referenced by a replica command that needs the file content. Files specified in the ArchiveDataFilter: field are transferred to the replica regardless of whether any users of the replica have made requests for their content. To automatically transfer files on submit, use the syntax: **// depot/pattern/…**

Type	Name	Description
		To exclude files from automatic transfer, use the syntax: `-//depot/pattern/…` All patterns are specified in depot syntax.
String	distributedConfig	For an edge or commit server, this optional field, which is displayed only when you use the -l or -c option, shows configuration settings for this server. `-l` flag shows the current configuration. `-c-` flag shows current configuration values, recommended default values for fields that are not set, or unset for fields that are not set and do not have default values. If this field is present when invoked with -c, the configuration commands in this field are run on the current server using the scope of the server specified in the serverID field.

StreamCommand

The Perforce stream specification defines a single stream.

Streams are hierarchical branches with policies that control the structure and the flow of change. Stream hierarchies are based on the stability of the streams, specified by the type you assign to the stream. Development streams are least stable (most subject to change), mainline streams are somewhat stable, and release streams are highly stable. Virtual streams can be used to copy and merge between parent and child streams without storing local data. Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data.

Stream contents are defined by the paths that you map. By default, a stream has the same structure as its parent (the stream from which it was branched), but you can override the structure, for example to ensure that specified files cannot be submitted or integrated to other streams.

Table 302. Properties

Type	Name	Description
String	stream	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form `//depotname/streamname`.
Date	update	The date the stream specification was last modified.
Date	access	The date and time that the stream specification was last accessed by any Perforce command.
String	owner	The Perforce user or group who owns the stream. The default is the user who created the stream.

Type	Name	Description
String	name	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname portion of the stream path.
String	parent	The parent of this stream. Must be none if the stream's `Type:` is mainline, otherwise must be set to an existing stream identifier of the form `//depotname/streamname`.
String	type	The stream's type determines the expected flow of change. Valid stream types are `mainline`, `virtual`, `development`, and `release`. • `mainline`: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream. • `virtual`: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream. • `release`: More stable than the mainline. Release streams copy from the parent and merge to the parent. • `development`: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent. • `task`: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.
String	description	Description of the stream.
String	options	Settings that configure stream behavior as follows: • `[un]locked`: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification

Type	Name	Description
		cannot be deleted, and only its owner can modify it. The default is unlocked.
		• `[all,owner]submit`: Specifies whether all users or only the owner of the stream can submit changes to the stream. The default is allsubmit. If the Owner: of a stream marked ownersubmit is a group, all users who are members of that group can submit changes to the stream.
		• `[no]toparent`: Specifies whether integrations from the stream to its parent are expected. The default is toparent.
		• `[no]fromparent`: Specifies whether integrations to the stream from its parent are expected. The default is fromparent for mainline and development streams, and nofromparent for release streams.
		• `mergeany,mergedown`: Specifies whether the merge flow is restricted or whether merge is permitted from any other stream. For example, the mergeany option would allow a merge from a child to a parent with no warnings. A virtual stream must have its flow options set to notoparent and nofromparent. Flow options are ignored for mainline streams.
Array of String	paths	Paths define how files are incorporated into the stream structure. Specify paths using the following format: `path_type view_path [depot_path]` where `path_type` is a single keyword, `view_path` is a file path with no leading slashes, and the optional `depot_path` is a file path beginning with `//`.
		The default path is `share` …
		Valid path types are:
		• `share view_path`: Specified files can be synced, submitted, and integrated to and from the parent stream.
		• `isolate view_path`: Specified files can be synced and submitted, but cannot be integrated to and from the parent stream.
		• `import view_path [depot_path]`: Specified files can be synced, but cannot be submitted or integrated to and from the parent stream. The `view_path` is mapped as in the parent stream's view, or to an (optional) `depot_path`. The `depot_path` may include a changelist specifier. That stream's client workspaces will be limited to seeing

Type	Name	Description
		revisions at that change or lower within that depot path. For example, you can specify a depot path like this: **//depot/import/…@1000**. Revisions from changelists greater than 1000 will be automatically hidden from most commands. The changelist limits in effect for a given stream workspace are displayed in a read-only client workspace specification field called ChangeView. • **import+ view_path [depot_path]**: Functions like a standard import path, enabling you to map a path from outside the stream depot to your stream, but unlike a standard import path, you can submit changes to the files in an import+ path. • **exclude view_path**: Specified files cannot be synced, submitted or integrated to and from the parent stream. By default, streams inherit their structure from the parent stream (except mainlines, which have no parent). Paths are inherited by child stream views; a child stream's path can downgrade the inherited view, but not upgrade it. (For example, a child stream can downgrade a shared path to an isolated path, but if the parent stream defines a path as isolated, its child cannot restore full access by specifying the path as shared.) Note that the **depot_path** is relevant only when the path_type is **import** or **import+**.
Array of String	remapped	Reassigns the location of workspace files. To specify the source path and its location in the workspace, use the following syntax: **view_path_1 view_path_2** where **view_path_1** and **view_path_2** are Perforce view paths (omit leading slashes and leading or embedded wildcards; terminal wildcards are fine). For example, to ensure that files are synced to the local ProjectX folder, remap as follows: … **projectX/…** Line ordering in the Remapped: field is significant: if more than one line remaps the same files, the later line takes precedence. Remappings are inherited by child streams and the workspaces associated with them.
Array of String	ignored	A list of file or directory names to be ignored in client views. For example: ``` /tmp # ignores files named "tmp" /tmp/... # ignores directories named "tmp" .tmp # ignores file names ending in .tmp ```

Type	Name	Description
		Lines in the `Ignored:` field can appear in any order. Ignored files and directories are inherited by child stream client views.

StreamsCommand

A summary of a stream in the system, as provided by the `p4 streams` command.

Table 303. Properties

Type	Name	Description
String	stream	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form `//depotname/streamname`.
Date	update	The date the stream specification was last modified.
Date	access	The date and time that the stream specification was last accessed by any Perforce command.
String	owner	The Perforce user or group who owns the stream. The default is the user who created the stream.
String	name	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname portion of the stream path.
String	parent	The parent of this stream. Must be none if the stream's `Type:` is mainline, otherwise must be set to an existing stream identifier of the form `//depotname/streamname`.
String	type	The stream's type determines the expected flow of change. Valid stream types are `mainline`, `virtual`, `development`, and `release`. • `mainline`: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream. • `virtual`: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream. • `release`: More stable than the mainline. Release streams copy from the parent and merge to the parent.

Type	Name	Description
		• `development`: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent. • `task`: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.
String	description	Description of the stream.
String	options	Settings that configure stream behavior as follows: • `[un]locked`: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification cannot be deleted, and only its owner can modify it. The default is unlocked. • `` `[all``

Triggers

Defines the triggers table, like it would appear in the output to the `p4 triggers` command.

Table 304. Properties

Type	Name	Description
Array of String	triggers	A list of trigger definitions. A trigger definition contains four fields that specify the name of the trigger, the type of event that should trigger the execution of the script, the paths that should be affected by the trigger, the location of the script, and other trigger type-dependent information. When the condition specified in a trigger definition is satisfied, the associated script or program is executed. Example: `trig1 change-submit //depot/dir/… "/usr/bin/s1.pl %changelist%"` See the Helix Versioning Engine Administrator Guide for more details on trigger definitions.

UserCommand

Create or edit Perforce user specifications and preferences.

There are three types of Perforce users: standard users, operator users, and service users. Standard users are the default, and each standard user consumes one Perforce license. The operator user type is intended for system administrators; they are subject to the same restrictions on permissions as any other user, but are further restricted in that they can run only a limited subset of Perforce commands. Service users are intended for inter-server communication in replicated and multi-server environments, and are restricted to an even smaller subset of Perforce commands. Neither operators nor service users consume Perforce licenses.

Table 305. Properties

Type	Name	Description
String	user	The Perforce username.
String	type	Type of user: standard, operator, or service. Once you set the type, you cannot change it.
String	authMethod	One of the following: perforce or ldap. Specifying perforce enables authentication using Perforce's internal db.user table or by way of an authentication trigger. This is the default unless it is overridden with the auth.default.method configurable. Specifying ldap enables authentication against AD/LDAP servers specified by the currently active LDAP configurations.
String	email	The user's email address. By default, this is user@client.
Date	update	The date and time this specification was last updated.
Date	access	The date and time this user last ran a Perforce command.
String	fullName	The user's full name.
String	jobView	Jobs matching this jobview appear on any changelists created by this user. Jobs that are fixed by the changelist should be left in the changelist when it's submitted with p4 submit; other jobs should be deleted from the form before submission.
String	password	The user's password.
Date	passwordChange	The date and time of the user's last password change. If the user has no password, this field is blank.

Type	Name	Description
Array of String	reviews	A list of files the user would like to review. This field can include exclusionary mappings.

UsersCommand

Table 306. Properties

Type	Name	Description
String	user	The Perforce username.
String	type	Type of user: standard, operator, or service. Once you set the type, you cannot change it.
String	email	The user's email address. By default, this is user@client.
Date	update	The date and time this specification was last updated.
Date	access	The date and time this user last ran a Perforce command.
String	fullName	The user's full name.
String	hasPassword	If 'enabled', the password has been set on the user.

PHP SDK Reference

Getting Started

Inside the `clients/php` directory of the installation is an SDK for using Helix Web Services from PHP.

Typically, you'll create an instance of `HelixWebServices\Api\DefaultApi`, configure it, and log in to your user.

```
use HelixWebServices\Api\DefaultApi;
use HelixWebServices\Model\LoginRequest;

$api = new DefaultApi();
$api->getApiClient()->getConfig()->setHost("https://helix.mycompany.com/api/2016.1.0");

// If you are using the default installation, you'll likely need to disable SSL verification.
// The default install creates a self-signed certificate, which is intended to be replaced
// by your certificate.
// $api->getApiClient()->getConfig()->setSslVerification(false);

$login_request = new LoginRequest();
$login_request->setUser("myuser");
$login_request->setPassword("mypassword");
$login_response = $api->loginPost($login_request);

$api->getApiClient()->getConfig()->setApiKey("Authorization", $login_response->getTicket());

// Now, list depots available on the p4d server with id 'myserver'
$depots = $api->serverDepotsGet("myserver");
```

> **Warning**
> The PHP client library includes the **/api/2016.1.0** base path as part of the host URL configuration. In most other clients, this is a separate configuration value.

ApiClient Reference

The **ApiClient** implements basic interacting with the server, that is used by **DefaultApi** and **AlphaApi** classes. You access this usually by the **getApiClient()** method of these classes.

```
$api = new HelixWebServices\Api\DefaultApi();
$api->getApiClient();
```

ApiClient::getConfig

Method Signature.

```
Configuration HelixWebServices\ApiClient::getConfig()
```

Table 307. Returns

Type	Notes
See "Configuration Reference" on page 194	

Configuration Reference

The **Configuration** class is used by the **ApiClient** as the pool of settings that alter behavior for interacting with the underlying server. It is created with the **ApiClient**, which is typically owned by the **DefaultApi** or **AlphaApi** instance you create.

```
$api = new HelixWebServices\Api\DefaultApi();
$config = $api->getApiClient()->getConfig();
```

Table 308. Properties

Name	Type	Description
apiKey	string	The authentication token for the user, as returned from "DefaultApi::loginPost" on page 197.
curlTimeout	string	The timeout (in seconds) of the HTTP request. Defaults to 0, which means no timeout.
debug	bool	Enable debug messaging. Defaults to false.
debugFile	string	Debug file location. Defaults to STDOUT by default.
host	string	The base URL we use to connect to the server, including the /api/VERSION string, e.g., https://helix.mycompany.com/api/2016.1.0.
tempFolderPath	string	Debug file location.
sslVerification	boolean	Set to false to disable SSL verification.
userAgent	string	The User-Agent of the HTTP request, set to "PHP-Swagger" by default.

Configuration::addDefaultHeader

Adds a default request header to all requests.

Method Signature.

```
Configuration Configuration::addDefaultHeader($headerName, $headerValue)
```

Table 309. Parameters

Name	Type	Description	Required
$headerName	string	The header name (e.g., Token).	Yes

Table 310. Returns

Type	Notes
"Configuration Reference" on page 194	This configuration object.

Configuration::deleteDefaultHeader

Removes a request header that may exist in the configuration.

Method Signature.

```
Configuration Configuration::deleteDefaultHeader($headerName)
```

Table 311. Parameters

Name	Type	Description	Required
$headerName	string	The header name (e.g., Token).	Yes

Table 312. Returns

Type	Notes
"Configuration Reference" on page 194	This configuration object.

Configuration::getDefaultHeaders

Retruns the array of request headers we set.

Method Signature.

```
array Configuration::getDefaultHeaders()
```

Table 313. Returns

Type	Notes
array	The mapping of default request header names to values.

DefaultApi Reference

DefaultApi::configP4dsGet

The list of registered p4d servers in your cluster.

This is provided by a special set of configuration files in the system. For more information, consult the Helix Web Services user guide.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::configP4dsGet()
```

Table 314. Returns

Type	Notes
array of ["P4dConfigId" on page 259](#)	

DefaultApi::loginPost

Logs into the primary authentication source.

This can either be a p4d instance or Helix Cloud, depending upon the configuration of your Helix Web Services instance.

Method Signature.

```
LoginResponse HelixWebServices\Api\DefaultApi::loginPost($loginRequest)
```

Table 315. Parameters

Name	Type	Description	Required
loginRequest	["LoginRequest" on](#)	The user login and password.	true

Table 316. Returns

Type	Notes
["LoginResponse" on page 259](#)	Object with ticket to use for Basic auth password.

DefaultApi::statusGet

A simple structure to monitor for "problems" an admin should take care of, and, report the current application version.

This method does not require authentication.

Method Signature.

```
HWSStatus HelixWebServices\Api\DefaultApi::statusGet()
```

Table 317. Returns

Type	Notes
["HWSStatus" on page 255](#)	

DefaultApi::serverBranchesGet

Lists available branches in the system. The resources of this list are summaries of branches in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverBranchesGet($server)
```

Table 318. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 319. Returns

Type	Notes
array of "BranchesCommand" on page 23	Summaries of branches in the system.

DefaultApi::serverBranchesPost

Creates a new branch specification, like the `p4 branch` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverBranchesPost($server, $body)
```

Table 320. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
body	"BranchCommand"	The branch specification.	true

Table 321. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverBranchesBranchGet

Returns the branch spec details of the particular branch.

Method Signature.

```
BranchCommand HelixWebServices\Api\DefaultApi::serverBranchesBranchGet($server, $branch)
```

Table 322. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
branch	string	The branch ID	true

Table 323. Returns

Type	Notes
"BranchCommand" on page 232	Branch spec details

DefaultApi::serverBranchesBranchPatch

Update branch specifications, similar to the `p4 branch` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverBranchesBranchPatch($server, $branch,
  $body)
```

Table 324. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
branch	string	The branch ID	true
body	"BranchCommand"	Fields of the branch to update	true

Table 325. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverBranchesBranchDelete

Removes the branch specification, similar to the `p4 branch -d` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverBranchesBranchDelete($server, $branch)
```

Table 326. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
branch	string	The branch ID	true

Table 327. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverChangesGet

Lists available changes in the system. The resources of this list are summaries of changes in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverChangesGet($server, $max, $status, $user, $files)
```

Table 328. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
max	int	Limit the number of change results	false
status	string	The status of the changes, e.g., `submitted`	false
user	string	The user's login who submitted the change	false
files	string	Limit changes to the depot path expressions. See the changes command description.	false

Table 329. Returns

Type	Notes
array of "ChangesCommand" on page 23	Summaries of changes in the system.

DefaultApi::serverChangesChangeGet

Returns the change spec details of the particular change.

Method Signature.

```
ChangeCommand HelixWebServices\Api\DefaultApi::serverChangesChangeGet($server, $change)
```

Table 330. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
change	string	The change ID	true

Table 331. Returns

Type	Notes
"ChangeCommand" on page 233	Change spec details

DefaultApi::serverClientsGet

Lists available clients in the system. The resources of this list are summaries of clients in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverClientsGet($server)
```

Table 332. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 333. Returns

Type	Notes
array of "ClientsCommand" on page 240	Summaries of clients in the system.

DefaultApi::serverClientsPost

Creates a new client specification, like the `p4 client` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverClientsPost($server, $client)
```

Table 334. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
client	"ClientCommand"	The client spec	true

Table 335. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverClientsClientGet

Returns the client spec details of the particular client.

Method Signature.

```
ClientCommand HelixWebServices\Api\DefaultApi::serverClientsClientGet($server, $client)
```

Table 336. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
client	string	The client ID	true

Table 337. Returns

Type	Notes
"ClientCommand" on page 236	Client spec details

DefaultApi::serverClientsClientPatch

Update client specifications, similar to the `p4 client` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverClientsClientPatch($server, $client,
  $body)
```

Table 338. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
client	string	The client ID	true
body	"ClientCommand"	Fields of the client to update	true

Table 339. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverClientsClientDelete

Removes the client specification, similar to the `p4 client -d` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverClientsClientDelete($server, $client)
```

Table 340. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
client	string	The client ID	true

Table 341. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverCommandsCommandGet

Execute a Perforce command that requires no input. This only allows commands that have been whitelisted on your system. See the "Configuration" on page 23 section for details.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverCommandsCommandGet($server, $command,
 $arg)
```

Table 342. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
command	string	The command name	true
arg		Command arguments	false

Table 343. Returns

Type	Notes
"CommandResponse" on page 2⁄	Generic list of hashes response

DefaultApi::serverCommandsCommandPost

Execute a Perforce command that accepts input, like a spec. This only allows commands that have been whitelisted on your system. See the "Configuration" on page 23 section for details.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverCommandsCommandPost($server, $command,
 $arg, $input)
```

Table 344. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
command	string	The command name	true
arg		Command arguments	false
input	"CommandRequest	A hash used as input to the command	false

Table 345. Returns

Type	Notes
"CommandResponse" on page 2⁄	Generic list of hashes response

DefaultApi::serverCountersGet

Lists available counters in the system. The resources of this list are summaries of counters in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverCountersGet($server)
```

Table 346. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 347. Returns

Type	Notes
array of "Counter" on page 243	Summaries of counters in the system.

DefaultApi::serverCountersCounterPut

Update counter specifications, similar to the `p4 counter` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverCountersCounterPut($server, $counter,
  $body)
```

Table 348. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
counter	string	The counter ID	true
body	"Counter" on page	Fields of the counter to update	true

Table 349. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverCountersCounterGet

Returns the counter spec details of the particular counter.

Method Signature.

```
Counter HelixWebServices\Api\DefaultApi::serverCountersCounterGet($server, $counter)
```

Table 350. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
counter	string	The counter ID	true

Table 351. Returns

Type	Notes
"Counter" on page 243	Counter spec details

DefaultApi::serverCountersCounterDelete

Removes the counter specification, similar to the `p4 counter -d` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverCountersCounterDelete($server, $counter)
```

Table 352. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
counter	string	The counter ID	true

Table 353. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverCountersCounterIncrementPost

Increments a numerical counter, similar to the `p4 counter -i` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverCountersCounterIncrementPost($server,
  $counter)
```

Table 354. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
counter	string	The counter ID	true

Table 355. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverDepotsGet

Lists available depots in the system. The resources of this list are summaries of depots in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverDepotsGet($server)
```

Table 356. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 357. Returns

Type	Notes
array of "DepotsCommand" on page 245	Summaries of depots in the system.

DefaultApi::serverDepotsPost

Creates a new depot specification, like the `p4 depot` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverDepotsPost($server, $depot)
```

Table 358. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
depot	"DepotCommand"	The depot spec	true

Table 359. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverDepotsDepotGet

Returns the depot spec details of the particular depot.

Method Signature.

```
DepotCommand HelixWebServices\Api\DefaultApi::serverDepotsDepotGet($server, $depot)
```

Table 360. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
depot	string	The depot ID	true

Table 361. Returns

Type	Notes
"DepotCommand" on page 243	Depot spec details

DefaultApi::serverDepotsDepotPatch

Update depot specifications, similar to the `p4 depot` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverDepotsDepotPatch($server, $depot, $body)
```

Table 362. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
depot	string	The depot ID	true
body	"DepotCommand"	Fields of the depot to update	true

Table 363. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverDepotsDepotDelete

Removes the depot specification, similar to the `p4 depot -d` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverDepotsDepotDelete($server, $depot)
```

Table 364. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
depot	string	The depot ID	true

Table 365. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverGroupsGet

Lists available groups in the system. The resources of this list are summaries of groups in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverGroupsGet($server)
```

Table 366. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 367. Returns

Type	Notes
array of ["GroupsCommand" on page 254](#)	Summaries of groups in the system.

DefaultApi::serverGroupsPost

Creates a new group specification, like the **p4 group** command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverGroupsPost($server, $body)
```

Table 368. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
body	["GroupCommand"](#)	The group spec	true

Table 369. Returns

Type	Notes
["CommandResponse" on page 24](#)	

DefaultApi::serverGroupsGroupGet

Returns the group spec details of the particular group.

Method Signature.

```
GroupCommand HelixWebServices\Api\DefaultApi::serverGroupsGroupGet($server, $group)
```

Table 370. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
group	string	The group ID	true

Table 371. Returns

Type	Notes
"GroupCommand" on page 253	Group spec details

DefaultApi::serverGroupsGroupPatch

Update group specifications, similar to the `p4 group` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverGroupsGroupPatch($server, $group, $body)
```

Table 372. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
group	string	The group ID	true
body	"GroupCommand"	Fields of the group to update	true

Table 373. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverGroupsGroupDelete

Removes the group specification, similar to the `p4 group -d` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverGroupsGroupDelete($server, $group)
```

Table 374. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
group	string	The group ID	true

Table 375. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverJobsGet

Lists available jobs in the system. The resources of this list are summaries of jobs in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverJobsGet($server)
```

Table 376. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 377. Returns

Type	Notes
array of "JobsCommand" on page 255	Summaries of jobs in the system.

DefaultApi::serverJobsPost

Creates a new job specification, like the `p4 job` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverJobsPost($server, $job)
```

Table 378. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
job	"JobCommand" on	The job spec	true

Table 379. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverJobsJobGet

Returns the job spec details of the particular job.

Method Signature.

```
JobCommand HelixWebServices\Api\DefaultApi::serverJobsJobGet($server, $job)
```

Table 380. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
job	string	The job ID	true

Table 381. Returns

Type	Notes
"JobCommand" on page 255	Job spec details

DefaultApi::serverJobsJobPatch

Update job specifications, similar to the `p4 job` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverJobsJobPatch($server, $job, $jobCommand)
```

Table 382. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
job	string	The job ID	true
jobCommand	["JobCommand" on](#)	Fields of the job to update	true

Table 383. Returns

Type	Notes
["CommandResponse" on page 24](#)	

DefaultApi::serverJobsJobDelete

Removes the job specification, similar to the `p4 job -d` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverJobsJobDelete($server, $job)
```

Table 384. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
job	string	The job ID	true

Table 385. Returns

Type	Notes
["CommandResponse" on page 24](#)	

DefaultApi::serverJobsJobFixesChangeDelete

Removes the fix record association for the job for a particular changelist.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverJobsJobFixesChangeDelete($server, $job,
  $change)
```

Table 386. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
job	string	The job ID	true
change	string	The change ID	true

Table 387. Returns

Type	Notes
[“CommandResponse” on page 24](#)	

DefaultApi::serverJobsJobFixesChangePost

Adds a fix record to the job for a particular changelist.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverJobsJobFixesChangePost($server, $job,
  $change, $status)
```

Table 388. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
job	string	The job ID	true
change	string	The change ID	true
status	string	Specify the job status instead of using the default. The default is typically `closed` or some other value defined in the `Presets` field specified in the `p4 jobspec` form. If the changelist to which you're linking the job been submitted, the status value is immediately reflected in the job's status. If the changelist is pending, the job status is changed on submission of the changelist, provided that the -s option	false

Name	Type	Description	Required
		is also supplied to p4 submit and the desired status appears next to the job in the p4 submit form's Jobs: field. To leave a job unchanged, use the special status of same.	

Table 389. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverLabelsGet

Lists available labels in the system. The resources of this list are summaries of labels in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverLabelsGet($server)
```

Table 390. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 391. Returns

Type	Notes
array of "LabelsCommand" on page 256	Summaries of labels in the system.

DefaultApi::serverLabelsPost

Creates a new label specification, like the `p4 label` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverLabelsPost($server, $label)
```

Table 392. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
label	"LabelCommand" (	The label spec	true

Table 393. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverLabelsLabelGet

Returns the label spec details of the particular label.

Method Signature.

```
LabelCommand HelixWebServices\Api\DefaultApi::serverLabelsLabelGet($server, $label)
```

Table 394. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
label	string	The label ID	true

Table 395. Returns

Type	Notes
"LabelCommand" on page 256	Label spec details

DefaultApi::serverLabelsLabelPatch

Update label specifications, similar to the `p4 label` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverLabelsLabelPatch($server, $label,
  $labelCommand)
```

Table 396. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
label	string	The label ID	true
labelCommand	"LabelCommand" (	Fields of the label to update	true

Table 397. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverLabelsLabelDelete

Removes the label specification, similar to the `p4 label -d` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverLabelsLabelDelete($server, $label)
```

Table 398. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
label	string	The label ID	true

Table 399. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverLoginPost

Logs into a Helix Versioning Engine (p4d) server.

Method Signature.

```
LoginResponse HelixWebServices\Api\DefaultApi::serverLoginPost($server, $body)
```

Table 400. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
body	"LoginRequest" on	The user login and password.	true

Table 401. Returns

Type	Notes
"LoginResponse" on page 259	Object with ticket to use for Basic auth password.

DefaultApi::serverPathsGet

Lists depots, files, and directories in the system. This combines the output of the `p4 depots`, `p4 dirs`, and `p4 files` commands, depending upon your path.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverPathsGet($server, $path)
```

Table 402. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
path	string	The path "under a depot" to query under, e.g., `//depot/main`. This will list the directories and files underneath that path.	false

Table 403. Returns

Type	Notes
array of "Location" on page 258	Array of depots.

DefaultApi::serverProtectionsPut

Updates the protections table.

This method requires superuser access.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverProtectionsPut($server, $protections)
```

Table 404. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
protections	[“Protections” on pa](#)	The new protections table	true

Table 405. Returns

Type	Notes
[“CommandResponse” on page 24](#)	

DefaultApi::serverProtectionsGet

Returns a list of available protections in the system. The elements of this list are rows of the system's protections table.

This method requires superuser access.

See the output of **p4 protect** for more information.

Method Signature.

```
Protections HelixWebServices\Api\DefaultApi::serverProtectionsGet($server)
```

Table 406. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 407. Returns

Type	Notes
[“Protections” on page 259](#)	Object including list of protections entries

DefaultApi::serverServersGet

Lists available servers in the system. The resources of this list are summaries of servers in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverServersGet($server)
```

Table 408. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 409. Returns

Type	Notes
array of ["ServersCommand" on page 261](#)	Summaries of servers in the system.

DefaultApi::serverServersPost

Creates a new server specification, like the **p4 server** command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverServersPost($server, $serverCommand)
```

Table 410. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
serverCommand	["ServerCommand"](#)	The server spec	true

Table 411. Returns

Type	Notes
["CommandResponse" on page 24](#)	

DefaultApi::serverServersServerIdGet

Returns the server spec details of the particular server.

Method Signature.

```
ServerCommand HelixWebServices\Api\DefaultApi::serverServersServerIdGet($server, $serverId)
```

Table 412. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
serverId	string	The server ID of the server spec	true

Table 413. Returns

Type	Notes
"ServerCommand" on page 262	Server spec details

DefaultApi::serverServersServerIdPatch

Update server specifications, similar to the `p4 server` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverServersServerIdPatch($server, $serverId,
  $serverCommand)
```

Table 414. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
serverId	string	The server ID	true
serverCommand	"ServerCommand"	Fields of the server to update	true

Table 415. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverServersServerIdDelete

Removes the server specification, similar to the `p4 server -d` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverServersServerIdDelete($server, $serverId)
```

Table 416. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
serverId	string	The server ID	true

Table 417. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverStreamsGet

Lists available streams in the system. The resources of this list are summaries of streams in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverStreamsGet($server)
```

Table 418. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 419. Returns

Type	Notes
array of "StreamsCommand" on page 269	Summaries of streams in the system.

DefaultApi::serverStreamsPost

Creates a new stream specification, like the `p4 stream` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverStreamsPost($server, $body)
```

Table 420. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
body	"StreamCommand"	The stream spec	true

Table 421. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverStreamsStreamGet

Returns the stream spec details of the particular stream.

Method Signature.

```
StreamCommand HelixWebServices\Api\DefaultApi::serverStreamsStreamGet($server, $stream)
```

Table 422. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
stream	string	The stream ID	true

Table 423. Returns

Type	Notes
"StreamCommand" on page 265	Stream spec details

DefaultApi::serverStreamsStreamPatch

Update stream specifications, similar to the `p4 stream` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverStreamsStreamPatch($server, $stream,
  $body)
```

Table 424. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
stream	string	The stream ID	true
body	"StreamCommand"	Fields of the stream to update	true

Table 425. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverStreamsStreamDelete

Removes the stream specification, similar to the `p4 stream -d` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverStreamsStreamDelete($server, $stream)
```

Table 426. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
stream	string	The stream ID	true

Table 427. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverTriggersPut

Updates the triggers table.

This method requires superuser access.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverTriggersPut($server, $triggers)
```

Table 428. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
triggers	"Triggers" on page	The new triggers table	true

Table 429. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverTriggersGet

Returns a list of available triggers in the system. The elements of this list are rows of the system's triggers table.

This method requires superuser access.

Method Signature.

```
Triggers HelixWebServices\Api\DefaultApi::serverTriggersGet($server)
```

Table 430. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 431. Returns

Type	Notes
"Triggers" on page 270	List of triggers entries

DefaultApi::serverUsersGet

Lists available users in the system. The resources of this list are summaries of users in the system.

Method Signature.

```
array HelixWebServices\Api\DefaultApi::serverUsersGet($server, $includeService, $max)
```

Table 432. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
includeService	bool	If true, shows service users in the list.	false
max	int	Cap the number of users reported to this amount.	false

Table 433. Returns

Type	Notes
array of "UsersCommand" on page 272	Summaries of users in the system.

DefaultApi::serverUsersPost

Creates a new user specification, like the `p4 user` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverUsersPost($server, $body)
```

Table 434. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
body	"UserCommand" o	The user spec	true

Table 435. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverUsersUserGet

Returns the user spec details of the particular user.

Method Signature.

```
UserCommand HelixWebServices\Api\DefaultApi::serverUsersUserGet($server, $user)
```

Table 436. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
user	string	The user ID	true

Table 437. Returns

Type	Notes
"UserCommand" on page 271	User spec details

DefaultApi::serverUsersUserPatch

Update user specifications, similar to the `p4 user` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverUsersUserPatch($server, $user, $body)
```

Table 438. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
user	string	The user ID	true
body	"UserCommand" o	Fields of the user to update	true

Table 439. Returns

Type	Notes
"CommandResponse" on page 24	

DefaultApi::serverUsersUserDelete

Removes the user specification, similar to the `p4 user -d` command.

Method Signature.

```
CommandResponse HelixWebServices\Api\DefaultApi::serverUsersUserDelete($server, $user)
```

Table 440. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
user	string	The user ID	true

Table 441. Returns

Type	Notes
"CommandResponse" on page 24	

AlphaApi Reference

AlphaApi::serverChangesPost

Create a new changelist that can affect multiple files using different kinds of actions. If you require the ability to integrate or move, for example, you can use this method.

Method Signature.

```
CommandResponse HelixWebServices\Api\AlphaApi::serverChangesPost($server, $changelistRequest)
```

Table 442. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
changelistRequest	"ChangelistRequest	Description of changes to make	true

Table 443. Returns

Type	Notes
"CommandResponse" on page 24	

AlphaApi::serverGitFusionReposGet

Lists all configured repositories readable by the current user. .Method Signature

```
array HelixWebServices\Api\AlphaApi::serverGitFusionReposGet($server)
```

Table 444. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true

Table 445. Returns

Type	Notes
array of "GitFusionRepoId" on page 248	List of configured repository names and IDs

AlphaApi::serverGitFusionReposPost

Submits a [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file to create or update a repository configuration.

If the repository does not exist or has been previously deleted, this method saves contents of the config file to a new **p4gf_config** file. If the repository has already been initialised, this method replaces all of the file contents of the specified repository's **p4gf_config** file.

Method Signature.

```
CommandResponse HelixWebServices\Api\AlphaApi::serverGitFusionReposPost($server, $body)
```

Table 446. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
body	"GitFusionRepoCor	The new configuration	true

Table 447. Returns

Type	Notes
"CommandResponse" on page 24	

AlphaApi::serverGitFusionReposRepoGet

Return configuration for the specified repository. Grabs and returns contents of the [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file for given repository.

Method Signature.

```
GitFusionRepoConfig HelixWebServices\Api\AlphaApi::serverGitFusionReposRepoGet($server, $repo)
```

Table 448. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
repo	string	The Git Fusion Repo ID	true

Table 449. Returns

Type	Notes
"GitFusionRepoConfig" on page	Git Fusion repository config

AlphaApi::serverGitFusionReposRepoPatch

Updates values in the repository configuration. This method will find all specified parameters and update each value for the specified repository's configuration file.

Method Signature.

```
CommandResponse HelixWebServices\Api\AlphaApi::serverGitFusionReposRepoPatch($server, $repo,
  $body)
```

Table 450. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
repo	string	The Git Fusion Repo ID	true
body	"GitFusionRepoCo	The new configuration	true

Table 451. Returns

Type	Notes
"CommandResponse" on page 24	

AlphaApi::serverGitFusionReposRepoDelete

Deletes the repository configuration (The [p4gf_config file](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j)). Contents of the repository are not deleted from Perforce.

Method Signature.

```
CommandResponse HelixWebServices\Api\AlphaApi::serverGitFusionReposRepoDelete($server, $repo)
```

Table 452. Parameters

Name	Type	Description	Required
server	string	The server ID that we execute this particular method against.	true
repo	string	The Git Fusion Repo ID	true

Table 453. Returns

Type	Notes
"CommandResponse" on page 24	

Model Reference

BranchCommand

Models the output of a `p4 branch` command.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_branch.html).

Table 454. Properties

Type	Name	Description
string	branch	The branch name, as provided on the command line.
string	owner	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.
\DateTime	access	The date the branch mapping was last accessed.
\DateTime	update	The date the branch mapping was last changed.
string	options	Either `unlocked` (the default) or `locked`. If `locked`, only the `Owner:` can modify the branch mapping, and the mapping can't be deleted until it is `unlocked`.
string	description	A short description of the branch's purpose.
Array of string	view	A set of mappings from one set of files in the depot (the source files) to another set of files in the depot (the target files). The view maps from one location in the depot to another; it can't refer to a client workspace.

Type	Name	Description
		For example, the branch view `//depot/main/... //depot/r2.1/...` maps all the files under `//depot/main` to `//depot/r2.1`.

BranchesCommand

A reference to a branch mapping known to the system.

Table 455. Properties

Type	Name	Description
string	branch	The branch name, as provided on the command line.
string	owner	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.
\DateTime	access	The date the branch mapping was last accessed.
\DateTime	update	The date the branch mapping was last changed.
string	options	Either `unlocked` (the default) or `locked`. If `locked`, only the `Owner:` can modify the branch mapping, and the mapping can't be deleted until it is `unlocked`.
string	description	A short description of the branch's purpose.

ChangeCommand

A changelist specification.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_change.html).

Table 456. Properties

Type	Name	Description
string	change	Contains the changelist number if editing an existing changelist, or `new` if creating a new changelist.
string	client	Name of current client workspace
\DateTime	date	Date the changelist was last modified.
string	user	Name of the change owner. The owner of an empty pending changelist (that is, a pending changelist without any files in it) can transfer

Type	Name	Description
		ownership of the changelist to another existing user either by editing this field, or by using the -U user option. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
string	status	`pending`, `shelved`, `submitted`, or `new`. Not editable by the user. The status is `new` when the changelist is created, `pending` when it has been created but has not yet been submitted to the depot, `shelved` when its contents are shelved, and `submitted` when its contents have been stored in the depot.
string	description	Textual description of changelist. If you do not have access to a restricted changelist, the description is replaced with a "no permission" message.
Array of string	jobs	A list of jobs that are fixed by this changelist.
string	type	Type of change: `restricted` or `public`. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change. Public changes are displayed without restrictions. By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.
Array of string	files	The list of files submitted in this changelist.
string	importedBy	Displays the name of the user who ran the p4 fetch, p4 push, or p4 unzip command that imported this change into the server. This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied.

Type	Name	Description
		In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.
string	identify	Contains a label which uniquely identifies this changelist across all servers where it has been fetched, pushed, or unzipped. This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied. In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.

ChangesCommand

Table 457. Properties

Type	Name	Description
string	change	The changelist ID
\DateTime	date	Last modification time of the changelist
string	user	The owner of the changelist
string	client	Name of current client workspace.
string	status	`pending`, `shelved`, `submitted`, or `new`. Not editable by the user. The status is `new` when the changelist is created, `pending` when it has been created but has not yet been submitted to the depot, `shelved` when its contents are shelved, and `submitted` when its contents have been stored in the depot.
string	type	Type of change: `restricted` or `public`. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change. Public changes are displayed without restrictions.

Type	Name	Description
		By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.
string	path	Depot paths affected by this changelist
string	description	Textual description of changelist. If you do not have access to a restricted changelist, the description is replaced with a "no permission" message.

ChangelistRequest

Table 458. Properties

Type	Name	Description
string	description	
string	stream	Optional stream ID to use in case you want to edit files in a stream.
Array of "ChangelistAction"	actions	

ChangelistAction

Table 459. Properties

Type	Name	Description
string	depotFile	The target file path to edit.
string	fromDepotFile	For "branch" or "move" actions, this indicates the source file location.
string	actionType	One of "upload", "branch", "move", or "delete"
string	content	Base64-encoded content
integer	requireVersion	If set, we will only operate if this is the current version of the file.

ClientCommand

The client workspace specification and its view.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html).

Table 460. Properties

Type	Name	Description
string	client	The client workspace name, as specified in the `P4CLIENT` environment variable or its equivalents.
string	owner	The name of the user who owns the workspace. The default is the user who created the workspace. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
\DateTime	update	The time the workspace specification was last modified.
\DateTime	access	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)
string	host	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host. The hostname must be provided exactly as it appears in the output of p4 info when run from that host. This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.
string	description	A textual description of the workspace. The default text is Created by owner.
string	root	The directory (on the local host) relative to which all the files in the `View:` are specified. The default is the current working directory. The path must be specified in local file system syntax. If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to map files to multiple drives.
Array of string	altRoots	Up to two optional alternate client workspace roots.

Type	Name	Description
		Perforce applications use the first of the main and alternate roots that match the application's current working directory. Use the p4 info command to display the root being used. This enables users to use the same Perforce client workspace specification on multiple platforms, even those with different directory naming conventions. If you are using multiple or alternate workspace roots (the AltRoots: field), you can always tell which root is in effect by looking at the Client root: reported by p4 info. If you are using a Windows directory in any of your workspace roots, you must specify the Windows directory as your main workspace root and specify your other workspace roots in the AltRoots: field.
string	options	A set of seven switches that control particular workspace options. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
string	submitOptions	Options to govern the default behavior of p4 submit. • submitunchanged All open files (with or without changes) are submitted to the depot. This is the default behavior of Perforce. • submitunchanged+reopen All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist. • revertunchanged Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted. • revertunchanged+reopen Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist.

Type	Name	Description
		• leaveunchanged Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist. • leaveunchanged+reopen Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged+reopen, except that no unchanged files are submitted to the depot.
string	lineEnd	Configure carriage-return/linefeed (CR/LF) conversion. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
string	stream	Associates the workspace with the specified stream. Perforce generates the view for stream-associated workspaces: you cannot modify it manually.
string	streamAtChange	A changelist number that sets a back-in-time view of a stream. When StreamAtChange is set, running p4 sync (when called with no arguments) updates the workspace to files at this changelist revision, instead of the head revision. You cannot submit changes (p4 submit returns an error) when StreamAtChange is set, because the workspace view no longer reflects the current stream inheritance. This field is ignored unless the Stream field is also set to a valid stream.
string	serverID	If set, restricts usage of the workspace to the named server. If unset, use is allowed on master server and on any replicas of the master other than Edge servers.
Array of string	view	Specifies the mappings between files in the depot and files in the workspace. A new view takes effect on the next p4 sync operation.
Array of string	changeView	Restricts access to depot paths to a particular point in time. Files specified for the ChangeView field are read-only: they

Type	Name	Description
		may be opened but not submitted. For example: **//depot/path/…@1000**
		Revisions of the files in the specified path will not be visible if they were submitted after the specified changelist number. Files matching a ChangeView path may not be submitted.
string	type	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts.
		Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.

ClientsCommand

Table 461. Properties

Type	Name	Description
string	client	The client workspace name, as specified in the **P4CLIENT** environment variable or its equivalents.
string	owner	The name of the user who owns the workspace. The default is the user who created the workspace.
		The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
\DateTime	update	The time the workspace specification was last modified.
\DateTime	access	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)
string	host	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host.

Type	Name	Description
		The hostname must be provided exactly as it appears in the output of p4 info when run from that host.
		This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.
string	description	A textual description of the workspace. The default text is Created by owner.
string	root	The directory (on the local host) relative to which all the files in the `View:` are specified. The default is the current working directory. The path must be specified in local file system syntax.
		If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to map files to multiple drives. additionalProperties:
string	type	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts.
		Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.
string	options	A set of seven switches that control particular workspace options.
		See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
string	submitOptions	Options to govern the default behavior of p4 submit. • submitunchanged

Type	Name	Description
		All open files (with or without changes) are submitted to the depot. This is the default behavior of Perforce.
		• submitunchanged+reopen
		All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist.
		• revertunchanged
		Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted.
		• revertunchanged+reopen
		Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist.
		• leaveunchanged
		Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist.
		• leaveunchanged+reopen
		Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged+reopen, except that no unchanged files are submitted to the depot.
string	lineEnd	Configure carriage-return/linefeed (CR/LF) conversion.
		See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
string	stream	Associates the workspace with the specified stream.
		Perforce generates the view for stream-associated workspaces: you cannot modify it manually.

CommandResponse

A generic container for responses from the p4d server that we have yet to completely classify.

Table 462. Properties

Type	Name	Description
Array of object	results	A collection of maps that have various values set by p4d.

CommandRequest

A single map typically defines input to generic command methods.

Table 463. Properties

Type	Name	Description
object	object	Don't use this. It's a kludge around a bug in the Java client code generator

Counter

A persistent variable in the server.

Table 464. Properties

Type	Name	Description
string	counter	The variable name
string	value	The variable value. Many variables are numerical in nature, which allow you to do atomic increment operations in method calls instead of having to fetch, increment, and save.

DepotCommand

The depot specification, which is the shared repository Perforce stores files in.

Table 465. Properties

Type	Name	Description
string	depot	The depot name.
string	owner	The user who owns the depot. By default, this is the user who created the depot. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does

Type	Name	Description
		not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
string	description	A short description of the depot's purpose. Optional.
string	type	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`.
		A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server.
		A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy.
		A remote depot references files that reside on other servers, and cannot be written to.
		The spec depot, if present, automatically archives edited forms.
		The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.
		An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries.
		A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.
string	address	If the `Type:` is `remote`, the address should be the P4PORT address of the remote server. If the `Type:` is local or spec, this field is ignored.
string	suffix	If the Type: is spec, this field holds an optional suffix for generated paths to objects in the spec depot.
		The default suffix is .p4s. You do not need a suffix to use the spec depot, but supplying a file extension to your Perforce server's versioned specs enables users of GUI client software to associate Perforce specifications with a preferred text editor. If the Type: is local or remote, this field is ignored.

Type	Name	Description
string	streamDepth	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following the depot name of a stream.] This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.
string	map	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the ... wildcard; for example, a local depot new might have a Map: of `new/...`. If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, `//depot/new/rel2/...`. This directory will be the root of the local representation of the remote depot.
Array of string	specMap	For spec depots, an optional description of which specs should be saved, expressed as a view.

DepotsCommand

A summary of depots in the system, with information provided by the `p4 depots` command.

Table 466. Properties

Type	Name	Description
string	depot	The depot name.
string	map	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the ... wildcard; for example, a local depot new might have a Map: of `new/...`. If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, `//depot/new/rel2/...`. This directory will be the root of the local representation of the remote depot.
string	type	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`. A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server.

Type	Name	Description
		A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy.
		A remote depot references files that reside on other servers, and cannot be written to.
		The spec depot, if present, automatically archives edited forms.
		The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.
		An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries.
		A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.
string	streamDepth	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following the depot name of a stream.]
		This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.
string	description	A short description of the depot's purpose. Optional.

DirsCommand

Table 467. Properties

Type	Name	Description
string	dir	

FilesCommand

Table 468. Properties

Type	Name	Description
string	depotFile	

Type	Name	Description
string	revision	
string	change	
string	action	
\DateTime	time	
string	type	

FstatCommand

Detailed information about each file, as provided by the **p4 fstat** command.

Table 469. Properties

Type	Name	Description
string	depotFile	Depot path to file. For files containing special characters, the filename is displayed containing the ASCII expression of the character's hexadecimal value.
string	movedFile	Name in depot of moved to/from file.
string	shelved	Set to **shelved** if file is shelved.
string	headAction	Action taken at head revision, if in depot. One of: add, edit, delete, branch, move/add, move/delete, integrate, import, purge, or archive.
string	headChange	Head revision changelist number, if in depot.
string	headRev	Head revision number, if in depot.
string	headType	Head revision type, if in depot.
string	headCharset	Head charset, for unicode files.
\DateTime	headTime	Head revision changelist time, if in depot. Time is measured in seconds since 00:00:00 UTC, January 1, 1970.
\DateTime	headModTime	Head revision modification time (the time that the file was last modified on the client before submit), if in depot.
string	movedRev	Head revision of moved file.
string	digest	MD5 digest of a file.
string	fileSize	File length in bytes.

Type	Name	Description
string	actionOwner	User who opened the file, if open.
string	resolved	The number, if any, of resovled integration records.
string	unresolved	The number, if any, of unresolved integration records.
string	reresolvable	The number, if any, of re-resolvable integration records.
Array of string	otherOpens	For each user with the file open, the workspace and user with the open file.
Array of string	otherLocks	For each user with the file locked, the workspace and user holding the lock.
Array of string	otherActions	For each user with the file open, the action taken.
Array of string	otherChanges	The changelist number with this file open.
Array of string	resolveActions	Pending integration action.
Array of string	resolveBaseFiles	Pending base files.
Array of string	resolveBaseRevs	Pending base revision numbers.
Array of string	resolveFromFiles	Pending from files.
Array of string	resolveStartFromRe	Pending starting revisions.
Array of string	resolveEndFromRe	Pending ending revisions.

GitFusionRepoId

Table 470. Properties

Type	Name	Description
string	id	An identifier for the repository that can be used safely within URL paths.
string	name	The repository name, which can be path-like.

GitFusionRepoConfig

Table 471. Properties

Type	Name	Description
string	name	The repository name, which can be path-like.

Type	Name	Description
string	description	Repo description returned by the @list command.
"GitFusionRepoGlc	globalOverrides	
Array of "GitFusionRepoBra	branches	

GitFusionRepoBranchConfig

Defines a unique Git Fusion branch.

Table 472. Properties

Type	Name	Description
string	gitBranchId	Alphanumeric ID for the git branch. *Do not change this value once this repo has been cloned.*
string	gitBranchName	Defines a name specified in a local repo for a Git branch. A valid Git branch name. Do not edit this value after you clone the repo.
Array of string	view	Defines a Perforce workspace view mapping that maps Perforce depot paths (left side) to Git work tree paths (right side). Correctly formed mapping syntax; must not include any Perforce stream or spec depots, and all depot paths on the right side must match exactly across all branch definitions. You can add and remove only certain types of Perforce branches from this view after you clone the repo.
string	stream	Defines a Perforce stream that maps to the Git branch. Provide a stream name using the syntax //streamdepot/mystream. A Git Fusion branch can be defined as a view or a stream but not both. If your branch is defined as stream, it can include only one stream.
string	readOnly	Prohibit git pushes that introduce commits to the branch.

GitFusionRepoGlobalOverrides

A list of per-repo settings that override global settings.

Table 473. Properties

Type	Name	Description
string	charset	Defines the default Unicode setting that Git Fusion applies to new repos. This setting is valid only when Git Fusion interacts with a Unicode-enabled Perforce server. (Defaults to `UTF-8`).
string	depotPathRepoCrea	Allow Git users to create new repos by pushing/pulling a git url which specifies a Perforce depot path. This is similar to creating a repo from a p4 client.
string	depotPathRepoCrea	Restrict which authenticated Git pushers are allowed to create new repos when depot-path-repo-creation-enable is enabled.
string	changeOwner	Defines whether Git Fusion assigns either the Git commit author or the Git pusher as the owner of a pushed change (submit).
string	enableGitBranchCre	Defines whether Git Fusion creates a new branch of Perforce depot file hierarchy for each copied branch of Git workspace history, including Git task branches as Git Fusion anonymous branches.
string	enableSwarmReviev	Permits branch creation for Swarm reviews, even when enable-git-branch-creation is disabled.
string	enableGitMergeCor	Defines whether Git Fusion copies merge commits and displays them in Perforce as integrations between Perforce branches.
string	enableGitSubmodu	Defines whether Git Fusion allows Git submodules to be pushed to Perforce.
string	ignoreAuthorPermi	Defines whether Git Fusion evaluates both the author's and pusher's Perforce write permissions during a push or evaluates only the pusher's permissions.
string	preflightCommit	Enables you to trigger pre-flight commit scripts that enforce local policy for Git pushes. This can be especially useful if you have Perforce submit triggers that could reject a push and damage the repository.
string	readPermissionChe	Enables you to require that Git clone, pull, or fetch requests check the Perforce protections table for the puller's read permission on the files being pulled.
string	gitMergeAvoidance	If the Perforce service includes any changelists submitted by Git Fusion 13.2 or earlier, you can prevent unnecessary

Type	Name	Description
		merge commits by setting this key to the number of the last changelist submitted before your site upgraded to a later version of Git Fusion.
string	jobLookup	Set the format for entering Perforce jobs in Git commit descriptions so that they are recognized by Git Fusion and appear in Perforce changelists as fixes. By default, job IDs whose string starts with "job" (as in job123456) are passed through to the changelist description and job field. Use this option if you want Git Fusion to recognize additional expressions, such as JIRA issue IDs.
string	depotBranchCreati	Allow Git users to create new fully-populated depot branches within Perforce.
string	depotBranchCreati	Restrict the authenticated Git pushers who are allowed to create new fully-populated depot branches, if depotBranchCreationEnable is enabled.
string	depotBranchCreati	Tell Git Fusion where to create new fully-populated depot branches, if depotBranchCreationEnable is enabled. Default path is `//depot/[repo]/[git_branch_name]`.
string	depotBranchCreati	Set how the depot path set in depotBranchCreationDepotPath should appear in Git. Enter a Perforce view specification that maps Perforce depot paths (left side) to Git work tree paths (right side). Perforce depot paths are relative to the root set in depotBranchCreationDepotPath. The default ... ... maps every file under the depotBranchCreationDepotPath root to Git. Right side paths must match the right side for every other branch already defined within a repo.
string	enableGitFindCopi	When Git reports a copy file action, store that action in Perforce as a p4 integ. Often set in tandem with enableGitFindRenames. No/Off/0%: Do not use Git's copy detection. Treat all possible file copy actions as p4 add actions. 1%-100%: Use Git's copy detection. Value passed to git diff-tree --find-copies=n. Git Fusion also adds --find-copies-harder whenever adding --find-copies.

Type	Name	Description
string	enableGitFindRena	When Git reports a rename (also called move) file action, store that in Perforce as a p4 move. Often set in tandem with enableGitFindCopies. `No`/`Off`/`0%`: Do not use Git's rename detection. Treat all possible file rename actions as independent p4 delete and p4 add actions. `1%-100%`: Use Git's rename detection. Value passed to git diff-tree --find-renames=n.
string	enableStreamImpor	Enables you to convert Perforce stream import paths to Git submodules when you clone a Git Fusion repository. If set to Yes, you must also set either httpUrl or sshUrl.
string	httpUrl	The URL used by Git to clone a repository from Git Fusion over HTTP. This property is required if you want to use Perforce stream import paths as git submodules and you use HTTP(S).
string	sshUrl	The "URL" used by Git to clone a repository from Git Fusion using SSH. This property is required if you want to use Perforce stream import paths as git submodules and you use SSH.
string	emailCaseSensitivit	Defines whether Git Fusion pays attention to case when matching Git user email addresses to Perforce user account email addresses during the authorization check.
string	authorSource	Defines the source that Git Fusion uses to identify the Perforce user associated with a Git push. Defaults to `git-email`. Use any one of the following values: • `git-email`: Use the email address of the Git author to look for a Perforce user account with the same email address. Git Fusion consults the p4gf_usermap file first, and if that fails to produce a match, it scans the Perforce user table. • `git-user`: Use the user.name field in the Git commit. This is the part of the author field before the email address. • `git-email-account`: Use the account portion of the Git author's email address. If the Git author's email value is `samwise@the_shire.com`, Git Fusion uses the Perforce account samwise.

Type	Name	Description
		You can also tell Git Fusion to iterate through multiple source types until it finds a matching Perforce account. Specify the source types in order of precedence, separated by commas. For example: git-user, git-email-account, git-email.
string	limitSpaceMb	Natural number representing the number of megabytes of disk space that can be consumed by any single repo. This value does not include the spaced consumed on the Perforce server.
string	limitCommitsRecei	Natural number representing the maximum number of commits allowed in a single push.
string	limitFilesReceived	Natural number representing the maximum number of files allowed in a single push.
string	limitMegabytesRec	Natural number representing the maximum number of megabytes allowed in a single push.

GroupCommand

Add or delete users from a group, or set the maxresults, maxscanrows, maxlocktime, and timeout limits for the members of a group.

Table 474. Properties

Type	Name	Description
string	group	The name of the group, as entered on the command line.
string	maxResults	The maximum number of results that members of this group can access from the service from a single command. The default value is `unset`.
string	maxScanRows	The maximum number of rows that members of this group can scan from the service from a single command. The default value is `unset`.
string	maxLockTime	The maximum length of time (in milliseconds) that any one operation can lock any database table when scanning data. The default value is `unset`.
string	maxOpenFiles	The maximum number of files that a member of a group can open using a single command.
string	timeout	The duration (in seconds) of the validity of a session ticket created by p4 login. The default value is 43,200 seconds (12 hours). To create a ticket that does not expire, set the Timeout: field to `unlimited`.

Type	Name	Description
string	passwordTimeout	The length of time (in seconds) for which passwords for users in this group remain valid. To disable password aging, use a value of unset.
string	ldapConfig	The LDAP configuration to use when populating the group's user list from an LDAP query.
string	ldapSearchQuery	The LDAP query used to identify the members of the group.
string	ldapUserAttribute	The LDAP attribute that represents the user's username.
Array of string	subgroups	Names of other Perforce groups. To add all users in a previously defined group to the group you're presently working with, include the group name in the Subgroups: field of the p4 group form. Note that user and group names occupy separate namespaces, and thus, groups and users can have the same names. Every member of any previously defined group you list in the Subgroups: field will be a member of the group you're now defining.

Array of string	owners	Names of other Perforce users.
		
		`Group owners without super access are permitted to administer this group, provided that they use the -a option.`
		
		`Group owners are not necessarily members of a group; if a group owner is to be a member of the group, the userid must also be added to the Users: field.`
		
		`The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.`

Array of string	users	The Perforce usernames of the group members.

GroupsCommand

A list of entries that can show the layout how users are associated with the different groups in the system.

Table 475. Properties

Type	Name	Description
string	user	

Type	Name	Description
string	group	
string	isSubGroup	
string	isOwner	
string	isUser	
string	maxResults	
string	maxScanRows	
string	maxLockTime	
string	maxOpenFiles	
string	timeout	
string	passTimeout	

HWSStatus

Table 476. Properties

Type	Name	Description
string	status	When "OK" the server should be considered to be operating normally
string	version	The version of Helix Web Services server.

JobCommand

A defect, enhancement request, or other job specification.

The actual fields in a job can be edited by a superuser in your system. The default set of fields in a system are Job, Status, User, Date, and Description.

Table 477. Properties

Type	Name	Description
string	Job	The job name.

JobsCommand

A summary of jobs in the system.

The actual fields in a job can be edited by a superuser in your system. The default set of fields in a system are Job, Status, User, Date, and Description. Fields in the output of this command may be missing if the superuser removed User, Status, Date, or Description.

Table 478. Properties

Type	Name	Description
string	Job	The job name.

LabelsCommand

Table 479. Properties

Type	Name	Description
string	label	The label name.
\DateTime	update	The date the label specification was last modified.
\DateTime	access	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)
string	owner	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label.

The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user. |
| string | options | Options to control behavior and storage location of labels.

• locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync.

• autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels. |
| string | description | An optional description of the label's purpose. |

LabelCommand

A label specification.

Labels can be either automatic or static. Automatic labels refer to the revisions provided in the View: and Revision: fields. Static labels refer only to those specific revisions tagged by the label by means of either the p4 labelsync or p4 tag commands.

Table 480. Properties

Type	Name	Description
string	label	The label name.
string	owner	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
\DateTime	update	The date the label specification was last modified.
\DateTime	access	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)
string	description	An optional description of the label's purpose.
string	options	Options to control behavior and storage location of labels. • locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync. • autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels.
string	revision	An optional revision specification for an automatic label. If you use the # character to specify a revision number, you must use quotes around it in order to ensure that the # is parsed as a revision specifier, and not as a comment field in the form.
Array of string	view	A list of depot files that can be tagged with this label. No files are actually tagged until `p4 labelsync` is invoked.

Type	Name	Description
		Unlike client views or branch views, which map one set of files to another, label views consist of a simple list of depot files.
string	serverID	If set, restricts usage of the label to the named server. If unset, this label may be used on any server.

Location

A consolidated mechanism for identifying something that generally has a path in the system.

Each location references either a depot, a dir, or a file.

Table 481. Properties

Type	Name	Description
string	depotPath	An absolute depot path specification.
"DepotsCommand"	depot	
"DirsCommand" or	dir	
"FilesCommand" o	file	
"FstatCommand" o	fstat	
string	content	If this location indicates a single file, this can be set with the Base64-encoded content of the file.

LoginRequest

Captures the login information we need for logging into either a p4d server or our "authentication source".

Table 482. Properties

Type	Name	Description
string	user	Usually the Perforce username
string	password	
Array of "ServerLoginReque	serverLogins	

ServerLoginRequest

Table 483. Properties

Type	Name	Description
string	id	The server's ID
string	user	
string	password	

LoginResponse

Either of our login methods return a ticket, which is then used as a password in a basic authentication scheme.

When this is returned from the explicit p4d login, this is a host unlocked ticket, acceptable for using with a local client.

Table 484. Properties

Type	Name	Description
string	ticket	

P4dConfigId

Identification of servers the Helix Web Services instance can connect to.

Table 485. Properties

Type	Name	Description
string	id	A simple string identifier (alphanumeric characters only, please)
string	name	A display string, not guaranteed to be unique
string	description	A simple textual description, for potential selection by clients.

Protections

Displays the information stored in the `p4 protect` command.

Table 486. Properties

Type	Name	Description
Array of string	protections	Each item in the protections array is a line in the protections table, and is split into five columns.

Type	Name	Description
		1. Access level or mode. One of the access levels list, read, open, write, admin, super, review; or one of the rights =read, =open, =write, and =branch,
		2. Either **user** or **group**, to indicate what's identified by this entry.
		3. The group name or user name. To grant permission to all users, use a wildcard with just an asterix symbol.
		4. The IP address of the client host.
		5. The depot file path, which can contain wildcards. To exclude this mapping from the permission set, use a dash - as the first character of this value.
		IPv6 addresses and IPv4 addresses are also supported. You can use the * wildcard to refer to all IP addresses, but only when you are not using CIDR notation.
		If you use the * wildcard with an IPv6 address, you must enclose the entire IPv6 address in square brackets. For example, [2001:db8:1:2:*] is equivalent to [2001:db8:1:2::]/64. Best practice is to use CIDR notation, surround IPv6 addresses with brackets, and to avoid the * wildcard.
		How the system forms host addresses depends on the setting of the dm.proxy.protects variable. By default, this variable is set to 1. This means that if the client host uses some intermediary (proxy, broker, replica) to access the server, the proxy- prefix is prepended to the client host address to indicate that the connection is not direct. If you specify proxy-* for the Host field, that will affect all connections made via proxies, brokers, and replicas. A value like proxy-10.0.0.5 identifies a client machine with an IP address of 10.0.0.5 that is connected to the server through an intermediary.
		Setting the dm.proxy.protects variable to 0, removes the proxy- prefix and allows you to write a single set of protection entries that apply both to directly-connected clients as well as to those that connect via an intermediary. This is more convenient but less secure if it matters that a connection is made using an intermediary. If you use this setting, all intermediaries must be at release 2012.1 or higher.

ServersCommand

Table 487. Properties

Type	Name	Description
string	serverID	A unique identifier for this server. This must match the contents of the server's `server.id` file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.
string	type	Server executable type. One of the following: `server`, `proxy`, `broker`, `identifier`, `admin`. Each type may offer one or more services, defined in the `services` property.
string	services	The `server` type server provides the following services: • standard - a standard Perforce server • replica - a read-only replica server • commit-server - central server in distributed installation • edge-server - node in distributed installation • forwarding-replica - a replica configured to forward commands that involve database writes to a master server • build-server - a replica that supports build automation and build farm integration • P4AUTH - a server that provides authentication • P4CHANGE - a server that provides change numbering • depot-master - commit-server with automated failover • depot-standby - standby replica of the depot-master • workspace-server - node in a cluster installation • standby - read-only replica server that uses p4 journalcopy • forwarding-standby - forwarding replica server that uses p4 journalcopy The `proxy` type server provides a p4p caching proxy.

Type	Name	Description
		The **broker** type server provides the following services: • broker - a p4broker process • workspace-router - routing broker for a cluster The services field for the **identifier** type server specifies the existence of the cluster, and has the value **cluster**. The name of the cluster is then drawn from the ServerID field. The **admin** type server provides the following services: • hxca-server - the admin server for a Helix cluster. • zookeeper-server - ZooKeeper server for a cluster
string	name	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.
string	address	The P4PORT used by this server.
string	description	An optional description for this server.
string	user	The service user name used by the server.

ServerCommand

The Perforce server specification describes the high-level configuration and intended usage of a Perforce server. For installations with only one Perforce server, the server specification is optional.

Table 488. Properties

Type	Name	Description
string	serverID	A unique identifier for this server. This must match the contents of the server's **server.id** file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.
string	type	Server executable type. One of the following: **server**, **proxy**, **broker**, **identifier**, **admin**. Each type may offer one or more services, defined in the **services** property.
string	services	The **server** type server provides the following services: • standard - a standard Perforce server

Type	Name	Description
		• replica - a read-only replica server
		• commit-server - central server in distributed installation
		• edge-server - node in distributed installation
		• forwarding-replica - a replica configured to forward commands that involve database writes to a master server
		• build-server - a replica that supports build automation and build farm integration
		• P4AUTH - a server that provides authentication
		• P4CHANGE - a server that provides change numbering
		• depot-master - commit-server with automated failover
		• depot-standby - standby replica of the depot-master
		• workspace-server - node in a cluster installation
		• standby - read-only replica server that uses p4 journalcopy
		• forwarding-standby - forwarding replica server that uses p4 journalcopy
		The `proxy` type server provides a p4p caching proxy.
		The `broker` type server provides the following services:
		• broker - a p4broker process
		• workspace-router - routing broker for a cluster
		The services field for the `identifier` type server specifies the existence of the cluster, and has the value `cluster`. The name of the cluster is then drawn from the ServerID field.
		The `admin` type server provides the following services:
		• hxca-server - the admin server for a Helix cluster.
		• zookeeper-server - ZooKeeper server for a cluster
string	name	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.

Type	Name	Description
string	address	The P4PORT used by this server.
string	externalAddress	For an edge server, this optional field specifies the external address used for connections to a commit server. This field must be set for the edge server to enable parallel submits in a federated environment.
string	description	An optional description for this server.
string	user	The service user name used by the server.
string	clientDataFilter	For a replica server, this optional field can contain one or more patterns describing how active client workspace metadata is to be filtered. Active client workspace data includes have lists, working records, and pending resolves. To include client data, use the syntax: **//client-pattern/…** To exclude client data, use the syntax: **-//client-pattern/…** All patterns are specified in client syntax.
string	revisionDataFilter	For a replica server, this optional field can contain one or more patterns describing how submitted revision metadata is to be filtered. Submitted revision data includes revision records, integration records, label contents, and the files listed in submitted changelists. To include depot data, use the syntax: To exclude depot data, use the syntax: -//depot/pattern/… All patterns are specified in depot syntax.
string	archiveDataFilter	For a replica server, this optional field can contain one or more patterns describing the policy for automatically scheduling the replication of file content. If this field is present, only those files described by the pattern are automatically transferred to the replica; other files are not transferred until they are referenced by a replica command that needs the file content. Files specified in the ArchiveDataFilter: field are transferred to the replica regardless of whether any users of the replica have made requests for their content. To automatically transfer files on submit, use the syntax: **//depot/pattern/…**

Type	Name	Description
		To exclude files from automatic transfer, use the syntax: `-//depot/pattern/…`
		All patterns are specified in depot syntax.
string	distributedConfig	For an edge or commit server, this optional field, which is displayed only when you use the -l or -c option, shows configuration settings for this server.
		`-l` flag shows the current configuration. `-c-` flag shows current configuration values, recommended default values for fields that are not set, or unset for fields that are not set and do not have default values.
		If this field is present when invoked with -c, the configuration commands in this field are run on the current server using the scope of the server specified in the serverID field.

StreamCommand

The Perforce stream specification defines a single stream.

Streams are hierarchical branches with policies that control the structure and the flow of change. Stream hierarchies are based on the stability of the streams, specified by the type you assign to the stream. Development streams are least stable (most subject to change), mainline streams are somewhat stable, and release streams are highly stable. Virtual streams can be used to copy and merge between parent and child streams without storing local data. Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data.

Stream contents are defined by the paths that you map. By default, a stream has the same structure as its parent (the stream from which it was branched), but you can override the structure, for example to ensure that specified files cannot be submitted or integrated to other streams.

Table 489. Properties

Type	Name	Description
string	stream	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form `//depotname/streamname`.
\DateTime	update	The date the stream specification was last modified.
\DateTime	access	The date and time that the stream specification was last accessed by any Perforce command.
string	owner	The Perforce user or group who owns the stream. The default is the user who created the stream.

Type	Name	Description
string	name	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname portion of the stream path.
string	parent	The parent of this stream. Must be none if the stream's `Type:` is mainline, otherwise must be set to an existing stream identifier of the form `//depotname/streamname`.
string	type	The stream's type determines the expected flow of change. Valid stream types are `mainline`, `virtual`, `development`, and `release`. • `mainline`: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream. • `virtual`: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream. • `release`: More stable than the mainline. Release streams copy from the parent and merge to the parent. • `development`: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent. • `task`: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.
string	description	Description of the stream.
string	options	Settings that configure stream behavior as follows: • `[un]locked`: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification

Type	Name	Description
		cannot be deleted, and only its owner can modify it. The default is unlocked.
		• `[all,owner]submit`: Specifies whether all users or only the owner of the stream can submit changes to the stream. The default is allsubmit. If the Owner: of a stream marked ownersubmit is a group, all users who are members of that group can submit changes to the stream.
		• `[no]toparent`: Specifies whether integrations from the stream to its parent are expected. The default is toparent.
		• `[no]fromparent`: Specifies whether integrations to the stream from its parent are expected. The default is fromparent for mainline and development streams, and nofromparent for release streams.
		• `mergeany,mergedown`: Specifies whether the merge flow is restricted or whether merge is permitted from any other stream. For example, the mergeany option would allow a merge from a child to a parent with no warnings. A virtual stream must have its flow options set to notoparent and nofromparent. Flow options are ignored for mainline streams.
Array of string	paths	Paths define how files are incorporated into the stream structure. Specify paths using the following format: `path_type view_path [depot_path]` where `path_type` is a single keyword, `view_path` is a file path with no leading slashes, and the optional `depot_path` is a file path beginning with `//`.
		The default path is `share` …
		Valid path types are:
		• `share view_path`: Specified files can be synced, submitted, and integrated to and from the parent stream.
		• `isolate view_path`: Specified files can be synced and submitted, but cannot be integrated to and from the parent stream.
		• `import view_path [depot_path]`: Specified files can be synced, but cannot be submitted or integrated to and from the parent stream. The `view_path` is mapped as in the parent stream's view, or to an (optional) `depot_path`. The `depot_path` may include a changelist specifier. That stream's client workspaces will be limited to seeing

Type	Name	Description
		revisions at that change or lower within that depot path. For example, you can specify a depot path like this: **//depot/import/…@1000**. Revisions from changelists greater than 1000 will be automatically hidden from most commands. The changelist limits in effect for a given stream workspace are displayed in a read-only client workspace specification field called ChangeView. • **import+ view_path [depot_path]**: Functions like a standard import path, enabling you to map a path from outside the stream depot to your stream, but unlike a standard import path, you can submit changes to the files in an import+ path. • **exclude view_path**: Specified files cannot be synced, submitted or integrated to and from the parent stream. By default, streams inherit their structure from the parent stream (except mainlines, which have no parent). Paths are inherited by child stream views; a child stream's path can downgrade the inherited view, but not upgrade it. (For example, a child stream can downgrade a shared path to an isolated path, but if the parent stream defines a path as isolated, its child cannot restore full access by specifying the path as shared.) Note that the **depot_path** is relevant only when the path_type is **import** or **import+**.
Array of string	remapped	Reassigns the location of workspace files. To specify the source path and its location in the workspace, use the following syntax: **view_path_1 view_path_2** where **view_path_1** and **view_path_2** are Perforce view paths (omit leading slashes and leading or embedded wildcards; terminal wildcards are fine). For example, to ensure that files are synced to the local ProjectX folder, remap as follows: … **projectX/**… Line ordering in the Remapped: field is significant: if more than one line remaps the same files, the later line takes precedence. Remappings are inherited by child streams and the workspaces associated with them.
Array of string	ignored	A list of file or directory names to be ignored in client views. For example: ```/tmp # ignores files named "tmp"/tmp/... # ignores directories named "tmp".tmp # ignores file names ending in .tmp```

Type	Name	Description
		Lines in the `Ignored:` field can appear in any order. Ignored files and directories are inherited by child stream client views.

StreamsCommand

A summary of a stream in the system, as provided by the `p4 streams` command.

Table 490. Properties

Type	Name	Description
string	stream	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form `//depotname/streamname`.
\DateTime	update	The date the stream specification was last modified.
\DateTime	access	The date and time that the stream specification was last accessed by any Perforce command.
string	owner	The Perforce user or group who owns the stream. The default is the user who created the stream.
string	name	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname portion of the stream path.
string	parent	The parent of this stream. Must be none if the stream's `Type:` is mainline, otherwise must be set to an existing stream identifier of the form `//depotname/streamname`.
string	type	The stream's type determines the expected flow of change. Valid stream types are `mainline`, `virtual`, `development`, and `release`. • `mainline`: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream. • `virtual`: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream. • `release`: More stable than the mainline. Release streams copy from the parent and merge to the parent.

Type	Name	Description
		• development: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent.
		• task: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.
string	description	Description of the stream.
string	options	Settings that configure stream behavior as follows: • [un]locked: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification cannot be deleted, and only its owner can modify it. The default is unlocked. • `[all

Triggers

Defines the triggers table, like it would appear in the output to the `p4 triggers` command.

Table 491. Properties

Type	Name	Description
Array of string	triggers	A list of trigger definitions. A trigger definition contains four fields that specify the name of the trigger, the type of event that should trigger the execution of the script, the paths that should be affected by the trigger, the location of the script, and other trigger type-dependent information. When the condition specified in a trigger definition is satisfied, the associated script or program is executed. Example: `trig1 change-submit //depot/dir/… "/usr/bin/s1.pl %changelist%"` See the Helix Versioning Engine Administrator Guide for more details on trigger definitions.

UserCommand

Create or edit Perforce user specifications and preferences.

There are three types of Perforce users: standard users, operator users, and service users. Standard users are the default, and each standard user consumes one Perforce license. The operator user type is intended for system administrators; they are subject to the same restrictions on permissions as any other user, but are further restricted in that they can run only a limited subset of Perforce commands. Service users are intended for inter-server communication in replicated and multi-server environments, and are restricted to an even smaller subset of Perforce commands. Neither operators nor service users consume Perforce licenses.

Table 492. Properties

Type	Name	Description
string	user	The Perforce username.
string	type	Type of user: standard, operator, or service. Once you set the type, you cannot change it.
string	authMethod	One of the following: perforce or ldap. Specifying perforce enables authentication using Perforce's internal db.user table or by way of an authentication trigger. This is the default unless it is overridden with the auth.default.method configurable. Specifying ldap enables authentication against AD/LDAP servers specified by the currently active LDAP configurations.
string	email	The user's email address. By default, this is user@client.
\DateTime	update	The date and time this specification was last updated.
\DateTime	access	The date and time this user last ran a Perforce command.
string	fullName	The user's full name.
string	jobView	Jobs matching this jobview appear on any changelists created by this user. Jobs that are fixed by the changelist should be left in the changelist when it's submitted with p4 submit; other jobs should be deleted from the form before submission.
string	password	The user's password.
\DateTime	passwordChange	The date and time of the user's last password change. If the user has no password, this field is blank.

Type	Name	Description
Array of string	reviews	A list of files the user would like to review. This field can include exclusionary mappings.

UsersCommand

Table 493. Properties

Type	Name	Description
string	user	The Perforce username.
string	type	Type of user: standard, operator, or service. Once you set the type, you cannot change it.
string	email	The user's email address. By default, this is user@client.
\DateTime	update	The date and time this specification was last updated.
\DateTime	access	The date and time this user last ran a Perforce command.
string	fullName	The user's full name.
string	hasPassword	If 'enabled', the password has been set on the user.

Python SDK Reference

Getting Started

Inside the `clients/python` directory of the installation is an SDK for using Helix Web Services from Python.

Typically, you'll create instances of `ApiClient` and `DefaultApi`, log in, then update configuration with your Perforce ticket.

```
from helix_web_services_client import ApiClient, DefaultApi, configuration
from helix_web_services_client.models import LoginRequest

# Your system will likely start by accessing a self-signed certificate.
# Do not use this in production.
configuration.verify_ssl = False

# The URL version is typically specified by python clients as part of the host address
api_client = ApiClient('https://myhws.example.com/api/2016.1.0')

default_api = DefaultApi(api_client)

# Log in and grab a ticket
login_request = LoginRequest()
login_request.user = 'myuser'
login_request.password = 'mypassword'
login_response = self.default_api.login_post(login_request)

# Use the ticket with HTTP Basic authentication
configuration.api_key['Authorization'] = login_response.ticket

# OK, now you can use the default_api with commands
depots = default_api.server_depots_get('myserver')
```

We do not publish a module, so you'll have to go the "personal PyPI" route. The following instructions work on Unices. For more information see [the online Python guide](#).

Create a helix_web_services_client.tar.gz:

```
cd INSTALL_DIR/clients/python/helix_web_services_client
tar czf helix_web_services_client.tar.gz *
```

Then, go up a directory and start the PyPI server in the `clients/python` directory:

```
cd INSTALL_DIR/clients/python
python -m SimpleHTTPServer 9000
```

Then, install the module into your python distribution:

```
pip install --extra-index-url=http://127.0.0.1:9000/ helix_web_services_client
```

ApiClient reference

The `ApiClient` can be created, though is not required, before creating the API interfaces like `DefaultApi` or `AlphaApi`.

Constructor

Creates the `ApiClient` instance, that contains logic used by each of the API interfaces. Will be created by default by those API objects, and is typically accessible via the `api.api_client` property.

Method Signature.

```
ApiClient helix_web_services_client.ApiClient(host=None, header_name=None, header_value=None)
```

Table 494. Parameters

Name	Type	Description	Required
host	str	The base path for the server to call.	No
header_name	str	A header to pass when making calls to the API.	No
header_value	str	A header value to pass when making calls to the API.	No

Properties

Name	Type	Description	host
str	The base path for the server to call.	user_agent	str

ApiClient.set_default_header

Adds a default header value to requests.

Method Signature.

```
helix_web_services_client.ApiClient.set_default_header(header_name, header_value)
```

Table 495. Parameters

Name	Type	Description	Required
header_name	str	A header to pass when making calls to the API.	Yes
header_value	str	A header value to pass when making calls to the API.	Yes

Configuration reference

Typically accessed via the global value `configuration`, sets various settings for the client interaction.

```
from helix_web_services_client import configuration

configuration.api_key['Authorization'] = '1245xwadf...'
```

Properties

Name	Type	Description	api_key
dict	Stores the API key for	cert_file	str

Name	Type	Description	api_key
	authorization, typically at the 'Authorization' header. For example, `configuration.api` `=` `login_response.ti`		
The client certificate file, defaults to None.	debug	bool	Enable debug logging, defaults to False
key_file	str	The client key file, defaults to None.	ssl_ca_cert
str	Customize the certificate authority file to verify the peer, defaults to None.	verify_ssl	bool

DefaultApi reference

DefaultApi#config_p4ds_get

The list of registered p4d servers in your cluster.

This is provided by a special set of configuration files in the system. For more information, consult the Helix Web Services user guide.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#config_p4ds_get()
```

Table 496. Returns

Type	Notes
list["P4dConfigId" on page 348]	

DefaultApi#login_post

Logs into the primary authentication source.

This can either be a p4d instance or Helix Cloud, depending upon the configuration of your Helix Web Services instance.

Method Signature.

```
LoginResponse helix_web_services_client.apis.DefaultApi#login_post(loginRequest)
```

Table 497. Parameters

Name	Type	Description	Required
loginRequest	LoginRequest	The user login and password.	true

Table 498. Returns

Type	Notes
"LoginResponse" on page 348	Object with ticket to use for Basic auth password.

DefaultApi#status_get

A simple structure to monitor for "problems" an admin should take care of, and, report the current application version.

This method does not require authentication.

Method Signature.

```
HWSStatus helix_web_services_client.apis.DefaultApi#status_get()
```

Table 499. Returns

Type	Notes
"HWSStatus" on page 343	

DefaultApi#server_branches_get

Lists available branches in the system. The resources of this list are summaries of branches in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_branches_get(server)
```

Table 500. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 501. Returns

Type	Notes
list["BranchesCommand" on pag	Summaries of branches in the system.

DefaultApi#server_branches_post

Creates a new branch specification, like the `p4 branch` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_branches_post(server, body)
```

Table 502. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
body	BranchCommand	The branch specification.	true

Table 503. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_branches_branch_get

Returns the branch spec details of the particular branch.

Method Signature.

```
BranchCommand helix_web_services_client.apis.DefaultApi#server_branches_branch_get(server,
 branch)
```

Table 504. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
branch	str	The branch ID	true

Table 505. Returns

Type	Notes
"BranchCommand" on page 310	Branch spec details

DefaultApi#server_branches_branch_patch

Update branch specifications, similar to the `p4 branch` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_branches_branch_patch(server,
  branch, body)
```

Table 506. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
branch	str	The branch ID	true
body	BranchCommand	Fields of the branch to update	true

Table 507. Returns

Type	Notes
["CommandResponse" on page 32](#)	

DefaultApi#server_branches_branch_delete

Removes the branch specification, similar to the `p4 branch -d` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_branches_branch_delete(server,
  branch)
```

Table 508. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
branch	str	The branch ID	true

Table 509. Returns

Type	Notes
["CommandResponse" on page 32](#)	

DefaultApi#server_changes_get

Lists available changes in the system. The resources of this list are summaries of changes in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_changes_get(server, **kwargs)
```

Table 510. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
max = int	int	Limit the number of change results	No, optional keyword argument
status = str	str	The status of the changes, e.g., submitted	No, optional keyword argument
user = str	str	The user's login who submitted the change	No, optional keyword argument
files = str	str	Limit changes to the depot path expressions. See the changes command description.	No, optional keyword argument

Table 511. Returns

Type	Notes
list["ChangesCommand" on page	Summaries of changes in the system.

DefaultApi#server_changes_change_get

Returns the change spec details of the particular change.

Method Signature.

```
ChangeCommand helix_web_services_client.apis.DefaultApi#server_changes_change_get(server, change)
```

Table 512. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
change	str	The change ID	true

Table 513. Returns

Type	Notes
"ChangeCommand" on page 312	Change spec details

DefaultApi#server_clients_get

Lists available clients in the system. The resources of this list are summaries of clients in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_clients_get(server)
```

Table 514. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 515. Returns

Type	Notes
list["ClientsCommand" on page ʒ	Summaries of clients in the system.

DefaultApi#server_clients_post

Creates a new client specification, like the `p4 client` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_clients_post(server, client)
```

Table 516. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
client	ClientCommand	The client spec	true

Table 517. Returns

Type	Notes
[“CommandResponse” on page 32](#)	

DefaultApi#server_clients_client_get

Returns the client spec details of the particular client.

Method Signature.

```
ClientCommand helix_web_services_client.apis.DefaultApi#server_clients_client_get(server, client)
```

Table 518. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
client	str	The client ID	true

Table 519. Returns

Type	Notes
[“ClientCommand” on page 316](#)	Client spec details

DefaultApi#server_clients_client_patch

Update client specifications, similar to the `p4 client` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_clients_client_patch(server,
  client, body)
```

Table 520. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
client	str	The client ID	true
body	ClientCommand	Fields of the client to update	true

Table 521. Returns

Type	Notes
[“CommandResponse” on page 32](#)	

DefaultApi#server_clients_client_delete

Removes the client specification, similar to the `p4 client -d` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_clients_client_delete(server,
  client)
```

Table 522. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
client	str	The client ID	true

Table 523. Returns

Type	Notes
[“CommandResponse” on page 32](#)	

DefaultApi#server_commands_command_get

Execute a Perforce command that requires no input. This only allows commands that have been whitelisted on your system. See the [“Configuration” on page 23](#) section for details.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_commands_command_get(server,
  command, **kwargs)
```

Table 524. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
command	str	The command name	true

Name	Type	Description	Required
`arg = `		Command arguments	No, optional keyword argument

Table 525. Returns

Type	Notes
[“CommandResponse” on page 3](#)	Generic list of hashes response

DefaultApi#server_commands_command_post

Execute a Perforce command that accepts input, like a spec. This only allows commands that have been whitelisted on your system. See the [“Configuration” on page 23](#) section for details.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_commands_command_post(server,
  command, input, **kwargs)
```

Table 526. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
command	str	The command name	true
input	CommandRequest	A hash used as input to the command	false
`arg = `		Command arguments	No, optional keyword argument

Table 527. Returns

Type	Notes
[“CommandResponse” on page 3](#)	Generic list of hashes response

DefaultApi#server_counters_get

Lists available counters in the system. The resources of this list are summaries of counters in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_counters_get(server)
```

Table 528. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 529. Returns

Type	Notes
list["Counter" on page 326]	Summaries of counters in the system.

DefaultApi#server_counters_counter_put

Update counter specifications, similar to the `p4 counter` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_counters_counter_put(server,
  counter, body)
```

Table 530. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
counter	str	The counter ID	true
body	Counter	Fields of the counter to update	true

Table 531. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_counters_counter_get

Returns the counter spec details of the particular counter.

Method Signature.

```
Counter helix_web_services_client.apis.DefaultApi#server_counters_counter_get(server, counter)
```

Table 532. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
counter	str	The counter ID	true

Table 533. Returns

Type	Notes
"Counter" on page 326	Counter spec details

DefaultApi#server_counters_counter_delete

Removes the counter specification, similar to the `p4 counter -d` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_counters_counter_delete(server,
  counter)
```

Table 534. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
counter	str	The counter ID	true

Table 535. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_counters_counter_increment_post

Increments a numerical counter, similar to the `p4 counter -i` command.

Method Signature.

```
CommandResponse
  helix_web_services_client.apis.DefaultApi#server_counters_counter_increment_post(server,
  counter)
```

Table 536. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
counter	str	The counter ID	true

Table 537. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_depots_get

Lists available depots in the system. The resources of this list are summaries of depots in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_depots_get(server)
```

Table 538. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 539. Returns

Type	Notes
list["DepotsCommand" on page `	Summaries of depots in the system.

DefaultApi#server_depots_post

Creates a new depot specification, like the **p4 depot** command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_depots_post(server, depot)
```

Table 540. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
depot	DepotCommand	The depot spec	true

Table 541. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_depots_depot_get

Returns the depot spec details of the particular depot.

Method Signature.

```
DepotCommand helix_web_services_client.apis.DefaultApi#server_depots_depot_get(server, depot)
```

Table 542. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
depot	str	The depot ID	true

Table 543. Returns

Type	Notes
"DepotCommand" on page 327	Depot spec details

DefaultApi#server_depots_depot_patch

Update depot specifications, similar to the `p4 depot` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_depots_depot_patch(server,
  depot, body)
```

Table 544. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Name	Type	Description	Required
depot	str	The depot ID	true
body	DepotCommand	Fields of the depot to update	true

Table 545. Returns

Type	Notes
[“CommandResponse” on page 32](#)	

DefaultApi#server_depots_depot_delete

Removes the depot specification, similar to the `p4 depot -d` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_depots_depot_delete(server,
  depot)
```

Table 546. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
depot	str	The depot ID	true

Table 547. Returns

Type	Notes
[“CommandResponse” on page 32](#)	

DefaultApi#server_groups_get

Lists available groups in the system. The resources of this list are summaries of groups in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_groups_get(server)
```

Table 548. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 549. Returns

Type	Notes
list[<u>"GroupsCommand" on page</u>	Summaries of groups in the system.

DefaultApi#server_groups_post

Creates a new group specification, like the `p4 group` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_groups_post(server, body)
```

Table 550. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
body	GroupCommand	The group spec	true

Table 551. Returns

Type	Notes
<u>"CommandResponse" on page 32</u>	

DefaultApi#server_groups_group_get

Returns the group spec details of the particular group.

Method Signature.

```
GroupCommand helix_web_services_client.apis.DefaultApi#server_groups_group_get(server, group)
```

Table 552. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
group	str	The group ID	true

Table 553. Returns

Type	Notes
<u>"GroupCommand" on page 340</u>	Group spec details

DefaultApi#server_groups_group_patch

Update group specifications, similar to the `p4 group` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_groups_group_patch(server,
  group, body)
```

Table 554. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
group	str	The group ID	true
body	GroupCommand	Fields of the group to update	true

Table 555. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_groups_group_delete

Removes the group specification, similar to the `p4 group -d` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_groups_group_delete(server,
  group)
```

Table 556. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
group	str	The group ID	true

Table 557. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_jobs_get

Lists available jobs in the system. The resources of this list are summaries of jobs in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_jobs_get(server)
```

Table 558. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 559. Returns

Type	Notes
list["JobsCommand" on page 344	Summaries of jobs in the system.

DefaultApi#server_jobs_post

Creates a new job specification, like the `p4 job` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_jobs_post(server, job)
```

Table 560. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
job	JobCommand	The job spec	true

Table 561. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_jobs_job_get

Returns the job spec details of the particular job.

Method Signature.

```
JobCommand helix_web_services_client.apis.DefaultApi#server_jobs_job_get(server, job)
```

Table 562. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
job	str	The job ID	true

Table 563. Returns

Type	Notes
"JobCommand" on page 343	Job spec details

DefaultApi#server_jobs_job_patch

Update job specifications, similar to the `p4 job` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_jobs_job_patch(server, job,
  jobCommand)
```

Table 564. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
job	str	The job ID	true
jobCommand	JobCommand	Fields of the job to update	true

Table 565. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_jobs_job_delete

Removes the job specification, similar to the `p4 job -d` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_jobs_job_delete(server, job)
```

Table 566. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
job	str	The job ID	true

Table 567. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_jobs_job_fixes_change_delete

Removes the fix record association for the job for a particular changelist.

Method Signature.

```
CommandResponse
  helix_web_services_client.apis.DefaultApi#server_jobs_job_fixes_change_delete(server, job,
  change)
```

Table 568. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
job	str	The job ID	true
change	str	The change ID	true

Table 569. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_jobs_job_fixes_change_post

Adds a fix record to the job for a particular changelist.

Method Signature.

```
CommandResponse
  helix_web_services_client.apis.DefaultApi#server_jobs_job_fixes_change_post(server, job, change,
  **kwargs)
```

Table 570. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
job	str	The job ID	true
change	str	The change ID	true
`status = str`	str	Specify the job status instead of using the default. The default is typically `closed` or some other value defined in the `Presets` field specified in the `p4 jobspec` form. If the changelist to which you're linking the job been submitted, the status value is immediately reflected in the job's status. If the changelist is pending, the job status is changed on submission of the changelist, provided that the -s option is also supplied to p4 submit and the desired status appears next to the job in the p4 submit form's Jobs: field. To leave a job unchanged, use the special status of same.	No, optional keyword argument

Table 571. Returns

Type	Notes
[_"CommandResponse" on page 32_](#)	

DefaultApi#server_labels_get

Lists available labels in the system. The resources of this list are summaries of labels in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_labels_get(server)
```

Table 572. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 573. Returns

Type	Notes
list["LabelsCommand" on page 3	Summaries of labels in the system.

DefaultApi#server_labels_post

Creates a new label specification, like the `p4 label` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_labels_post(server, label)
```

Table 574. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
label	LabelCommand	The label spec	true

Table 575. Returns

Type	Notes
"CommandResponse" on page 3	

DefaultApi#server_labels_label_get

Returns the label spec details of the particular label.

Method Signature.

```
LabelCommand helix_web_services_client.apis.DefaultApi#server_labels_label_get(server, label)
```

Table 576. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
label	str	The label ID	true

Table 577. Returns

Type	Notes
"LabelCommand" on page 345	Label spec details

DefaultApi#server_labels_label_patch

Update label specifications, similar to the `p4 label` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_labels_label_patch(server,
 label, labelCommand)
```

Table 578. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
label	str	The label ID	true
labelCommand	LabelCommand	Fields of the label to update	true

Table 579. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_labels_label_delete

Removes the label specification, similar to the `p4 label -d` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_labels_label_delete(server,
 label)
```

Table 580. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
label	str	The label ID	true

Table 581. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_login_post

Logs into a Helix Versioning Engine (p4d) server.

Method Signature.

```
LoginResponse helix_web_services_client.apis.DefaultApi#server_login_post(server, body)
```

Table 582. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
body	LoginRequest	The user login and password.	true

Table 583. Returns

Type	Notes
"LoginResponse" on page 348	Object with ticket to use for Basic auth password.

DefaultApi#server_paths_get

Lists depots, files, and directories in the system. This combines the output of the `p4 depots`, `p4 dirs`, and `p4 files` commands, depending upon your path.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_paths_get(server, **kwargs)
```

Table 584. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
path = str	str	The path "under a depot" to query under, e.g., `//depot/main`. This will list the directories and files underneath that path.	No, optional keyword argument

Table 585. Returns

Type	Notes
list["Location" on page 347]	Array of depots.

DefaultApi#server_protections_put

Updates the protections table.

This method requires superuser access.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_protections_put(server,
  protections)
```

Table 586. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
protections	Protections	The new protections table	true

Table 587. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_protections_get

Returns a list of available protections in the system. The elements of this list are rows of the system's protections table.

This method requires superuser access.

See the output of **p4 protect** for more information.

Method Signature.

```
Protections helix_web_services_client.apis.DefaultApi#server_protections_get(server)
```

Table 588. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 589. Returns

Type	Notes
"Protections" on page 349	Object including list of protections entries

DefaultApi#server_servers_get

Lists available servers in the system. The resources of this list are summaries of servers in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_servers_get(server)
```

Table 590. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 591. Returns

Type	Notes
list["ServersCommand" on page	Summaries of servers in the system.

DefaultApi#server_servers_post

Creates a new server specification, like the `p4 server` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_servers_post(server,
  serverCommand)
```

Table 592. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
serverCommand	ServerCommand	The server spec	true

Table 593. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_servers_serverid_get

Returns the server spec details of the particular server.

Method Signature.

```
ServerCommand helix_web_services_client.apis.DefaultApi#server_servers_serverid_get(server,
  serverId)
```

Table 594. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
serverId	str	The server ID of the server spec	true

Table 595. Returns

Type	Notes
"ServerCommand" on page 353	Server spec details

DefaultApi#server_servers_serverid_patch

Update server specifications, similar to the `p4 server` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_servers_serverid_patch(server,
  serverId, serverCommand)
```

Table 596. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
serverId	str	The server ID	true
serverCommand	ServerCommand	Fields of the server to update	true

Table 597. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_servers_serverid_delete

Removes the server specification, similar to the `p4 server -d` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_servers_serverid_delete(server,
  serverId)
```

Table 598. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
serverId	str	The server ID	true

Table 599. Returns

Type	Notes
[“CommandResponse” on page 32](#)	

DefaultApi#server_streams_get

Lists available streams in the system. The resources of this list are summaries of streams in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_streams_get(server)
```

Table 600. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 601. Returns

Type	Notes
list[[“StreamsCommand” on page](#)	Summaries of streams in the system.

DefaultApi#server_streams_post

Creates a new stream specification, like the `p4 stream` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_streams_post(server, body)
```

Table 602. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
body	StreamCommand	The stream spec	true

Table 603. Returns

Type	Notes
[“CommandResponse” on page 32](#)	

DefaultApi#server_streams_stream_get

Returns the stream spec details of the particular stream.

Method Signature.

```
StreamCommand helix_web_services_client.apis.DefaultApi#server_streams_stream_get(server,
  **kwargs)
```

Table 604. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
stream = str	str	The stream ID	No, optional keyword argument

Table 605. Returns

Type	Notes
[“StreamCommand” on page 357](#)	Stream spec details

DefaultApi#server_streams_stream_patch

Update stream specifications, similar to the `p4 stream` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_streams_stream_patch(server,
  body, **kwargs)
```

Table 606. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
body	StreamCommand	Fields of the stream to update	true
`stream = str`	str	The stream ID	No, optional keyword argument

Table 607. Returns

Type	Notes
[“CommandResponse” on page 32](#)	

DefaultApi#server_streams_stream_delete

Removes the stream specification, similar to the `p4 stream -d` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_streams_stream_delete(server,
  **kwargs)
```

Table 608. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
`stream = str`	str	The stream ID	No, optional keyword argument

Table 609. Returns

Type	Notes
[“CommandResponse” on page 32](#)	

DefaultApi#server_triggers_put

Updates the triggers table.

This method requires superuser access.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_triggers_put(server, triggers)
```

Table 610. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
triggers	Triggers	The new triggers table	true

Table 611. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_triggers_get

Returns a list of available triggers in the system. The elements of this list are rows of the system's triggers table.

This method requires superuser access.

Method Signature.

```
Triggers helix_web_services_client.apis.DefaultApi#server_triggers_get(server)
```

Table 612. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 613. Returns

Type	Notes
"Triggers" on page 366	List of triggers entries

DefaultApi#server_users_get

Lists available users in the system. The resources of this list are summaries of users in the system.

Method Signature.

```
list helix_web_services_client.apis.DefaultApi#server_users_get(server, **kwargs)
```

Table 614. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
includeService = bool	bool	If true, shows service users in the list.	No, optional keyword argument
max = int	int	Cap the number of users reported to this amount.	No, optional keyword argument

Table 615. Returns

Type	Notes
list["UsersCommand" on page 36	Summaries of users in the system.

DefaultApi#server_users_post

Creates a new user specification, like the `p4 user` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_users_post(server, body)
```

Table 616. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
body	UserCommand	The user spec	true

Table 617. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_users_user_get

Returns the user spec details of the particular user.

Method Signature.

```
UserCommand helix_web_services_client.apis.DefaultApi#server_users_user_get(server, user)
```

Table 618. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
user	str	The user ID	true

Table 619. Returns

Type	Notes
"UserCommand" on page 366	User spec details

DefaultApi#server_users_user_patch

Update user specifications, similar to the `p4 user` command. Only the specified parameters in the body will be changed.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_users_user_patch(server, user,
  body)
```

Table 620. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
user	str	The user ID	true
body	UserCommand	Fields of the user to update	true

Table 621. Returns

Type	Notes
"CommandResponse" on page 32	

DefaultApi#server_users_user_delete

Removes the user specification, similar to the `p4 user -d` command.

Method Signature.

```
CommandResponse helix_web_services_client.apis.DefaultApi#server_users_user_delete(server, user)
```

Table 622. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
user	str	The user ID	true

Table 623. Returns

Type	Notes
"CommandResponse" on page 32	

AlphaApi reference

AlphaApi#server_changes_post

Create a new changelist that can affect multiple files using different kinds of actions. If you require the ability to integrate or move, for example, you can use this method.

Method Signature.

```
CommandResponse helix_web_services_client.apis.AlphaApi#server_changes_post(server,
  changelistRequest)
```

Table 624. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
changelistRequest	ChangelistRequest	Description of changes to make	true

Table 625. Returns

Type	Notes
"CommandResponse" on page 32	

AlphaApi#server_git_fusion_repos_get

Lists all configured repositories readable by the current user. .Method Signature

```
list helix_web_services_client.apis.AlphaApi#server_git_fusion_repos_get(server)
```

Table 626. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true

Table 627. Returns

Type	Notes
list["GitFusionRepoId" on page 3	List of configured repository names and IDs

AlphaApi#server_git_fusion_repos_post

Submits a [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file to create or update a repository configuration.

If the repository does not exist or has been previously deleted, this method saves contents of the config file to a new **p4gf_config** file. If the repository has already been initialised, this method replaces all of the file contents of the specified repository's **p4gf_config** file.

Method Signature.

```
CommandResponse helix_web_services_client.apis.AlphaApi#server_git_fusion_repos_post(server,
body)
```

Table 628. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
body	GitFusionRepoCon:	The new configuration	true

Table 629. Returns

Type	Notes
"CommandResponse" on page 3.	

AlphaApi#server_git_fusion_repos_repo_get

Return configuration for the specified repository. Grabs and returns contents of the [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file for given repository.

Method Signature.

```
GitFusionRepoConfig
 helix_web_services_client.apis.AlphaApi#server_git_fusion_repos_repo_get(server, repo)
```

Table 630. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
repo	str	The Git Fusion Repo ID	true

Table 631. Returns

Type	Notes
"GitFusionRepoConfig" on page	Git Fusion repository config

AlphaApi#server_git_fusion_repos_repo_patch

Updates values in the repository configuration. This method will find all specified parameters and update each value for the specified repository's configuration file.

Method Signature.

```
CommandResponse
 helix_web_services_client.apis.AlphaApi#server_git_fusion_repos_repo_patch(server, repo, body)
```

Table 632. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
repo	str	The Git Fusion Repo ID	true
body	GitFusionRepoCon:	The new configuration	true

Table 633. Returns

Type	Notes
"CommandResponse" on page 32	

AlphaApi#server_git_fusion_repos_repo_delete

Deletes the repository configuration (The [p4gf_config file](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j)). Contents of the repository are not deleted from Perforce.

Method Signature.

```
CommandResponse
  helix_web_services_client.apis.AlphaApi#server_git_fusion_repos_repo_delete(server, repo)
```

Table 634. Parameters

Name	Type	Description	Required
server	str	The server ID that we execute this particular method against.	true
repo	str	The Git Fusion Repo ID	true

Table 635. Returns

Type	Notes
"CommandResponse" on page 32	

Models

BranchCommand

Models the output of a `p4 branch` command.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_branch.html).

f .Attributes

Name	Type	Description
branch	str	The branch name, as provided on the command line.
owner	str	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.
access	datetime	The date the branch mapping was last accessed.
update	datetime	The date the branch mapping was last changed.
options	str	Either `unlocked` (the default) or `locked`.

Name	Type	Description
		If **locked**, only the **Owner:** can modify the branch mapping, and the mapping can't be deleted until it is **unlocked**.
description	str	A short description of the branch's purpose.
view	list[str]	A set of mappings from one set of files in the depot (the source files) to another set of files in the depot (the target files). The view maps from one location in the depot to another; it can't refer to a client workspace. `For example, the branch view //depot/main/... //depot/r2.1/... maps all the files under //depot/main to //depot/r2.1.`

BranchesCommand

A reference to a branch mapping known to the system.

f .Attributes

Name	Type	Description
branch	str	The branch name, as provided on the command line.
owner	str	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.
access	datetime	The date the branch mapping was last accessed.
update	datetime	The date the branch mapping was last changed.
options	str	Either **unlocked** (the default) or **locked**. If **locked**, only the **Owner:** can modify the branch mapping,

Name	Type	Description
		and the mapping can't be deleted until it is **unlocked**.
description	str	A short description of the branch's purpose.

ChangeCommand

A changelist specification.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_change.html).

f .Attributes

Name	Type	Description
change	str	Contains the changelist number if editing an existing changelist, or **new** if creating a new changelist.
client	str	Name of current client workspace
date	datetime	Date the changelist was last modified.
user	str	Name of the change owner. The owner of an empty pending changelist (that is, a pending changelist without any files in it) can transfer ownership of the changelist to another existing user either by editing this field, or by using the -U user option. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
status	str	**pending**, **shelved**, **submitted**, or **new**. Not editable by the user.

Name	Type	Description
		The status is `new` when the changelist is created, `pending` when it has been created but has not yet been submitted to the depot, `shelved` when its contents are shelved, and `submitted` when its contents have been stored in the depot.
description	str	Textual description of changelist. If you do not have access to a restricted changelist, the description is replaced with a "no permission" message.
jobs	list[str]	A list of jobs that are fixed by this changelist.
type	str	Type of change: `restricted` or `public`. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change. Public changes are displayed without restrictions. By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.
files	list[str]	The list of files submitted in this changelist.
importedBy	str	Displays the name of the user who ran the p4 fetch, p4 push, or p4 unzip command that

Name	Type	Description
		imported this change into the server.
		This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied.
		In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.
identify	str	Contains a label which uniquely identifies this changelist across all servers where it has been fetched, pushed, or unzipped.
		This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied.
		In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.

ChangesCommand

f .Attributes

Name	Type	Description
change	str	The changelist ID
date	datetime	Last modification time of the changelist

Name	Type	Description
user	str	The owner of the changelist
client	str	Name of current client workspace.
status	str	`pending`, `shelved`, `submitted`, or `new`. Not editable by the user. The status is `new` when the changelist is created, `pending` when it has been created but has not yet been submitted to the depot, `shelved` when its contents are shelved, and `submitted` when its contents have been stored in the depot.
type	str	Type of change: `restricted` or `public`. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change. Public changes are displayed without restrictions. By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.
path	str	Depot paths affected by this changelist
description	str	Textual description of changelist. If you do not have access to a restricted changelist, the

Name	Type	Description
		description is replaced with a "no permission" message.

ChangelistRequest

f .Attributes

Name	Type	Description
description	str	
stream	str	Optional stream ID to use in case you want to edit files in a stream.
actions	list["ChangelistAction" on page 3	

ChangelistAction

f .Attributes

Name	Type	Description
depotFile	str	The target file path to edit.
fromDepotFile	str	For "branch" or "move" actions, this indicates the source file location.
actionType	str	One of "upload", "branch", "move", or "delete"
content	str	Base64-encoded content
requireVersion	int	If set, we will only operate if this is the current version of the file.

ClientCommand

The client workspace specification and its view.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html).

f .Attributes

Name	Type	Description
client	str	The client workspace name, as specified in the P4CLIENT

Name	Type	Description
		environment variable or its equivalents.
owner	str	The name of the user who owns the workspace. The default is the user who created the workspace. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
update	datetime	The time the workspace specification was last modified.
access	datetime	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)
host	str	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host. The hostname must be provided exactly as it appears in the output of p4 info when run from that host. This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the

Name	Type	Description
		IP address restriction feature of p4 protect.
description	str	A textual description of the workspace. The default text is Created by owner.
root	str	The directory (on the local host) relative to which all the files in the **View:** are specified. The default is the current working directory. The path must be specified in local file system syntax.
		If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to map files to multiple drives.
altRoots	list[str]	Up to two optional alternate client workspace roots.
		Perforce applications use the first of the main and alternate roots that match the application's current working directory. Use the p4 info command to display the root being used.
		This enables users to use the same Perforce client workspace specification on multiple platforms, even those with different directory naming conventions.
		If you are using multiple or alternate workspace roots (the AltRoots: field), you can always tell which root is in effect by looking at the Client root: reported by p4 info.
		If you are using a Windows directory in any of your

Name	Type	Description
		workspace roots, you must specify the Windows directory as your main workspace root and specify your other workspace roots in the AltRoots: field.
options	str	A set of seven switches that control particular workspace options. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
submitOptions	str	Options to govern the default behavior of p4 submit. • submitunchanged All open files (with or without changes) are submitted to the depot. This is the default behavior of Perforce. • submitunchanged+reopen All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist. • revertunchanged Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted. • revertunchanged+reopen Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default

Name	Type	Description
		changelist. Unchanged files are reverted and not reopened in the default changelist.
		• leaveunchanged
		Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist.
		• leaveunchanged+reopen
		Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged+reopen, except that no unchanged files are submitted to the depot.
lineEnd	str	Configure carriage-return/linefeed (CR/LF) conversion. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
stream	str	Associates the workspace with the specified stream. Perforce generates the view for stream-associated workspaces: you cannot modify it manually.
streamAtChange	str	A changelist number that sets a back-in-time view of a stream. When StreamAtChange is set, running p4 sync (when called with no arguments)

Name	Type	Description
		updates the workspace to files at this changelist revision, instead of the head revision. You cannot submit changes (p4 submit returns an error) when StreamAtChange is set, because the workspace view no longer reflects the current stream inheritance. This field is ignored unless the Stream field is also set to a valid stream.
serverID	str	If set, restricts usage of the workspace to the named server. If unset, use is allowed on master server and on any replicas of the master other than Edge servers.
view	list[str]	Specifies the mappings between files in the depot and files in the workspace. A new view takes effect on the next p4 sync operation.
changeView	list[str]	Restricts access to depot paths to a particular point in time. Files specified for the ChangeView field are read-only: they may be opened but not submitted. For example: **//depot/path/…@1000** Revisions of the files in the specified path will not be visible if they were submitted after the specified changelist number. Files matching a ChangeView path may not be submitted.
type	str	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but

Name	Type	Description
		this should not be an issue in build scripts.
		Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.

ClientsCommand

f .Attributes

Name	Type	Description
client	str	The client workspace name, as specified in the **P4CLIENT** environment variable or its equivalents.
owner	str	The name of the user who owns the workspace. The default is the user who created the workspace. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
update	datetime	The time the workspace specification was last modified.
access	datetime	The date and time that the workspace was last used in any way. (Note: Reloading a

Name	Type	Description
		workspace with p4 reload does not affect the access time.)
host	str	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host. The hostname must be provided exactly as it appears in the output of p4 info when run from that host. This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.
description	str	A textual description of the workspace. The default text is Created by owner.
root	str	The directory (on the local host) relative to which all the files in the **View:** are specified. The default is the current working directory. The path must be specified in local file system syntax. If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to

Name	Type	Description
		map files to multiple drives. additionalProperties:
type	str	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts.
		Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.
options	str	A set of seven switches that control particular workspace options.
		See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
submitOptions	str	Options to govern the default behavior of p4 submit.
		• submitunchanged
		All open files (with or without changes) are submitted to the depot. This is the default behavior of Perforce.
		• submitunchanged+reopen

Name	Type	Description
		All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist. • revertunchanged Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted. • revertunchanged+reopen Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist. • leaveunchanged Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist. • leaveunchanged+reopen Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged+reopen, except that no unchanged files are submitted to the depot.

Name	Type	Description
lineEnd	str	Configure carriage-return/ linefeed (CR/LF) conversion. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
stream	str	Associates the workspace with the specified stream. Perforce generates the view for stream-associated workspaces: you cannot modify it manually.

CommandResponse

A generic container for responses from the p4d server that we have yet to completely classify.

f .Attributes

Name	Type	Description
results	list[object]	A collection of maps that have various values set by p4d.

CommandRequest

A single map typically defines input to generic command methods.

f .Attributes

Name	Type	Description
object	object	Don't use this. It's a kludge around a bug in the Java client code generator

Counter

A persistent variable in the server.

f .Attributes

Name	Type	Description
counter	str	The variable name
value	str	The variable value. Many variables are numerical in

Name	Type	Description
		nature, which allow you to do atomic increment operations in method calls instead of having to fetch, increment, and save.

DepotCommand

The depot specification, which is the shared repository Perforce stores files in.

f .Attributes

Name	Type	Description
depot	str	The depot name.
owner	str	The user who owns the depot. By default, this is the user who created the depot. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
description	str	A short description of the depot's purpose. Optional.
type	str	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`. A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server. A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy. A remote depot references files that reside on other servers, and cannot be written to.

Name	Type	Description
		The spec depot, if present, automatically archives edited forms.
		The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.
		An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries.
		A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.
address	str	If the `Type:` is `remote`, the address should be the P4PORT address of the remote server. If the `Type:` is local or spec, this field is ignored.
suffix	str	If the Type: is spec, this field holds an optional suffix for generated paths to objects in the spec depot. The default suffix is .p4s. You do not need a suffix to use the spec depot, but supplying a file extension to your Perforce server's versioned specs enables users of GUI client software to associate Perforce specifications with a preferred text editor. If the Type: is local or remote, this field is ignored.

Name	Type	Description
streamDepth	str	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following the depot name of a stream.] This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.
map	str	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the ... wildcard; for example, a local depot new might have a Map: of new/.... If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, //depot/new/rel2/.... This directory will be the root of the local representation of the remote depot.
specMap	list[str]	For spec depots, an optional description of which specs should be saved, expressed as a view.

DepotsCommand

A summary of depots in the system, with information provided by the `p4 depots` command.

f .Attributes

Name	Type	Description
depot	str	The depot name.
map	str	If the Type: is local, spec, or archive, set the map to point to the relative location of the

Name	Type	Description
		depot subdirectory. The map must contain the … wildcard; for example, a local depot new might have a Map: of **new/…**.
		If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, **//depot/new/rel2/…**. This directory will be the root of the local representation of the remote depot.
type	str	**local**, **remote**, **spec**, **stream**, **unload**, **archive** or **tangent**.
		A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server.
		A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy.
		A remote depot references files that reside on other servers, and cannot be written to.
		The spec depot, if present, automatically archives edited forms.
		The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.
		An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries.

Name	Type	Description
		A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.
streamDepth	str	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following the depot name of a stream.] This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.
description	str	A short description of the depot's purpose. Optional.

DirsCommand

f .Attributes

Name	Type	Description
dir	str	

FilesCommand

f .Attributes

Name	Type	Description
depotFile	str	
revision	str	
change	str	
action	str	
time	datetime	
type	str	

FstatCommand

Detailed information about each file, as provided by the `p4 fstat` command.

f .Attributes

Name	Type	Description
depotFile	str	Depot path to file. For files containing special characters, the filename is displayed containing the ASCII expression of the character's hexadecimal value.
movedFile	str	Name in depot of moved to/from file.
shelved	str	Set to **shelved** if file is shelved.
headAction	str	Action taken at head revision, if in depot. One of: add, edit, delete, branch, move/add, move/delete, integrate, import, purge, or archive.
headChange	str	Head revision changelist number, if in depot.
headRev	str	Head revision number, if in depot.
headType	str	Head revision type, if in depot.
headCharset	str	Head charset, for unicode files.
headTime	datetime	Head revision changelist time, if in depot. Time is measured in seconds since 00:00:00 UTC, January 1, 1970.
headModTime	datetime	Head revision modification time (the time that the file was last modified on the client before submit), if in depot.
movedRev	str	Head revision of moved file.
digest	str	MD5 digest of a file.
fileSize	str	File length in bytes.

Name	Type	Description
actionOwner	str	User who opened the file, if open.
resolved	str	The number, if any, of resovled integration records.
unresolved	str	The number, if any, of unresolved integration records.
reresolvable	str	The number, if any, of re-resolvable integration records.
otherOpens	list[str]	For each user with the file open, the workspace and user with the open file.
otherLocks	list[str]	For each user with the file locked, the workspace and user holding the lock.
otherActions	list[str]	For each user with the file open, the action taken.
otherChanges	list[str]	The changelist number with this file open.
resolveActions	list[str]	Pending integration action.
resolveBaseFiles	list[str]	Pending base files.
resolveBaseRevs	list[str]	Pending base revision numbers.
resolveFromFiles	list[str]	Pending from files.
resolveStartFromRevs	list[str]	Pending starting revisions.
resolveEndFromRevs	list[str]	Pending ending revisions.

GitFusionRepoId

f .Attributes

Name	Type	Description
id	str	An identifier for the repository that can be used safely within URL paths.
name	str	The repository name, which can be path-like.

GitFusionRepoConfig

f .Attributes

Name	Type	Description
name	str	The repository name, which can be path-like.
description	str	Repo description returned by the @list command.
globalOverrides	"GitFusionRepoGlobalOverrides"	
branches	list["GitFusionRepoBranchConfig	

GitFusionRepoBranchConfig

Defines a unique Git Fusion branch.

f .Attributes

Name	Type	Description
gitBranchId	str	Alphanumeric ID for the git branch. *Do not change this value once this repo has been cloned.*
gitBranchName	str	Defines a name specified in a local repo for a Git branch. A valid Git branch name. Do not edit this value after you clone the repo.
view	list[str]	Defines a Perforce workspace view mapping that maps Perforce depot paths (left side) to Git work tree paths (right side). Correctly formed mapping syntax; must not include any Perforce stream or spec depots, and all depot paths on the right side must match exactly across all branch definitions. You can add and remove only certain types of Perforce branches from this view after you clone the repo.

Name	Type	Description
stream	str	Defines a Perforce stream that maps to the Git branch. Provide a stream name using the syntax //streamdepot/mystream. A Git Fusion branch can be defined as a view or a stream but not both. If your branch is defined as stream, it can include only one stream.
readOnly	str	Prohibit git pushes that introduce commits to the branch.

GitFusionRepoGlobalOverrides

A list of per-repo settings that override global settings.

f .Attributes

Name	Type	Description
charset	str	Defines the default Unicode setting that Git Fusion applies to new repos. This setting is valid only when Git Fusion interacts with a Unicode-enabled Perforce server. (Defaults to UTF-8).
depotPathRepoCreationEnable	str	Allow Git users to create new repos by pushing/pulling a git url which specifies a Perforce depot path. This is similar to creating a repo from a p4 client.
depotPathRepoCreationP4group	str	Restrict which authenticated Git pushers are allowed to create new repos when depot-path-repo-creation-enable is enabled.
changeOwner	str	Defines whether Git Fusion assigns either the Git commit author or the Git pusher as the owner of a pushed change (submit).

Name	Type	Description
enableGitBranchCreation	str	Defines whether Git Fusion creates a new branch of Perforce depot file hierarchy for each copied branch of Git workspace history, including Git task branches as Git Fusion anonymous branches.
enableSwarmReviews	str	Permits branch creation for Swarm reviews, even when enable-git-branch-creation is disabled.
enableGitMergeCommits	str	Defines whether Git Fusion copies merge commits and displays them in Perforce as integrations between Perforce branches.
enableGitSubmodules	str	Defines whether Git Fusion allows Git submodules to be pushed to Perforce.
ignoreAuthorPermissions	str	Defines whether Git Fusion evaluates both the author's and pusher's Perforce write permissions during a push or evaluates only the pusher's permissions.
preflightCommit	str	Enables you to trigger pre-flight commit scripts that enforce local policy for Git pushes. This can be especially useful if you have Perforce submit triggers that could reject a push and damage the repository.
readPermissionCheck	str	Enables you to require that Git clone, pull, or fetch requests check the Perforce protections table for the puller's read permission on the files being pulled.
gitMergeAvoidanceAfterChange	str	If the Perforce service includes any changelists submitted by Git Fusion 13.2 or earlier,

Name	Type	Description
		you can prevent unnecessary merge commits by setting this key to the number of the last changelist submitted before your site upgraded to a later version of Git Fusion.
jobLookup	str	Set the format for entering Perforce jobs in Git commit descriptions so that they are recognized by Git Fusion and appear in Perforce changelists as fixes. By default, job IDs whose string starts with "job" (as in job123456) are passed through to the changelist description and job field. Use this option if you want Git Fusion to recognize additional expressions, such as JIRA issue IDs.
depotBranchCreationEnable	str	Allow Git users to create new fully-populated depot branches within Perforce.
depotBranchCreationP4group	str	Restrict the authenticated Git pushers who are allowed to create new fully-populated depot branches, if depotBranchCreationEnable is enabled.
depotBranchCreationDepotPath	str	Tell Git Fusion where to create new fully-populated depot branches, if depotBranchCreationEnable is enabled. Default path is **//depot/[repo]/[git_branch_name]**.
depotBranchCreationView	str	Set how the depot path set in depotBranchCreationDepotPath should appear in Git. Enter a Perforce view specification that maps Perforce depot paths (left side) to Git work tree paths (right

Name	Type	Description
		side). Perforce depot paths are relative to the root set in depotBranchCreationDepotPath. The default … … maps every file under the depotBranchCreationDepotPath root to Git. Right side paths must match the right side for every other branch already defined within a repo.
enableGitFindCopies	str	When Git reports a copy file action, store that action in Perforce as a p4 integ. Often set in tandem with enableGitFindRenames. No/Off/0%: Do not use Git's copy detection. Treat all possible file copy actions as p4 add actions. 1%-100%: Use Git's copy detection. Value passed to git diff-tree --find-copies=n. Git Fusion also adds --find-copies-harder whenever adding --find-copies.
enableGitFindRenames	str	When Git reports a rename (also called move) file action, store that in Perforce as a p4 move. Often set in tandem with enableGitFindCopies. `No`/`Off`/`0%`: Do not use Git's rename detection. Treat all possible file rename actions as independent p4 delete and p4 add actions. `1%`-`100%`: Use Git's rename detection. Value passed to git diff-tree --find-renames=n.
enableStreamImports	str	Enables you to convert Perforce stream import paths to Git submodules when you clone

Name	Type	Description
		a Git Fusion repository. If set to Yes, you must also set either httpUrl or sshUrl.
httpUrl	str	The URL used by Git to clone a repository from Git Fusion over HTTP. This property is required if you want to use Perforce stream import paths as git submodules and you use HTTP(S).
sshUrl	str	The "URL" used by Git to clone a repository from Git Fusion using SSH. This property is required if you want to use Perforce stream import paths as git submodules and you use SSH.
emailCaseSensitivity	str	Defines whether Git Fusion pays attention to case when matching Git user email addresses to Perforce user account email addresses during the authorization check.
authorSource	str	Defines the source that Git Fusion uses to identify the Perforce user associated with a Git push. Defaults to `git-email`. Use any one of the following values: • `git-email`: Use the email address of the Git author to look for a Perforce user account with the same email address. Git Fusion consults the p4gf_usermap file first, and if that fails to produce a match, it scans the Perforce user table. • `git-user`: Use the user.name field in the Git commit. This

Name	Type	Description
		is the part of the author field before the email address. • `git-email-account`: Use the account portion of the Git author's email address. If the Git author's email value is `samwise@the_shire.com`, Git Fusion uses the Perforce account samwise. You can also tell Git Fusion to iterate through multiple source types until it finds a matching Perforce account. Specify the source types in order of precedence, separated by commas. For example: git-user, git-email-account, git-email.
limitSpaceMb	str	Natural number representing the number of megabytes of disk space that can be consumed by any single repo. This value does not include the spaced consumed on the Perforce server.
limitCommitsReceived	str	Natural number representing the maximum number of commits allowed in a single push.
limitFilesReceived	str	Natural number representing the maximum number of files allowed in a single push.
limitMegabytesReceived	str	Natural number representing the maximum number of megabytes allowed in a single push.

GroupCommand

Add or delete users from a group, or set the maxresults, maxscanrows, maxlocktime, and timeout limits for the members of a group.

f .Attributes

Name	Type	Description
group	str	The name of the group, as entered on the command line.
maxResults	str	The maximum number of results that members of this group can access from the service from a single command. The default value is `unset`.
maxScanRows	str	The maximum number of rows that members of this group can scan from the service from a single command. The default value is `unset`.
maxLockTime	str	The maximum length of time (in milliseconds) that any one operation can lock any database table when scanning data. The default value is `unset`.
maxOpenFiles	str	The maximum number of files that a member of a group can open using a single command.
timeout	str	The duration (in seconds) of the validity of a session ticket created by p4 login. The default value is 43,200 seconds (12 hours). To create a ticket that does not expire, set the Timeout: field to `unlimited`.
passwordTimeout	str	The length of time (in seconds) for which passwords for users in this group remain valid. To disable password aging, use a value of unset.
ldapConfig	str	The LDAP configuration to use when populating the group's user list from an LDAP query.
ldapSearchQuery	str	The LDAP query used to identify the members of the group.
ldapUserAttribute	str	The LDAP attribute that represents the user's username.

Name	Type	Description
subgroups	list[str]	Names of other Perforce groups.
		To add all users in a previously defined group to the group you're presently working with, include the group name in the Subgroups: field of the p4 group form. Note that user and group names occupy separate namespaces, and thus, groups and users can have the same names.
		Every member of any previously defined group you list in the Subgroups: field will be a member of the group you're now defining.
owners	list[str]	Names of other Perforce users.
		`Group owners without super access are permitted to administer this group, provided that they use the -a option.`
		`Group owners are not necessarily members of a group; if a group owner is to be a member of the group, the userid must also be added to the Users: field.`
		`The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.`
users	list[str]	The Perforce usernames of the group members.

GroupsCommand

A list of entries that can show the layout how users are associated with the different groups in the system.

f .Attributes

Name	Type	Description
user	str	
group	str	
isSubGroup	str	
isOwner	str	
isUser	str	
maxResults	str	
maxScanRows	str	
maxLockTime	str	
maxOpenFiles	str	
timeout	str	
passTimeout	str	

HWSStatus

f .Attributes

Name	Type	Description
status	str	When "OK" the server should be considered to be operating normally
version	str	The version of Helix Web Services server.

JobCommand

A defect, enhancement request, or other job specification.

The actual fields in a job can be edited by a superuser in your system. The default set of fields in a system are Job, Status, User, Date, and Description.

f .Attributes

Name	Type	Description
Job	str	The job name.

JobsCommand

A summary of jobs in the system.

The actual fields in a job can be edited by a superuser in your system. The default set of fields in a system are Job, Status, User, Date, and Description. Fields in the output of this command may be missing if the superuser removed User, Status, Date, or Description.

f .Attributes

Name	Type	Description
Job	str	The job name.

LabelsCommand

f .Attributes

Name	Type	Description
label	str	The label name.
update	datetime	The date the label specification was last modified.
access	datetime	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)
owner	str	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
options	str	Options to control behavior and storage location of labels.

Name	Type	Description
		• locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync.
		• autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels.
description	str	An optional description of the label's purpose.

LabelCommand

A label specification.

Labels can be either automatic or static. Automatic labels refer to the revisions provided in the View: and Revision: fields. Static labels refer only to those specific revisions tagged by the label by means of either the p4 labelsync or p4 tag commands.

f .Attributes

Name	Type	Description
label	str	The label name.
owner	str	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.

Name	Type	Description
update	datetime	The date the label specification was last modified.
access	datetime	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)
description	str	An optional description of the label's purpose.
options	str	Options to control behavior and storage location of labels. • locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync. • autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels.
revision	str	An optional revision specification for an automatic label. If you use the # character to specify a revision number, you must use quotes around it in order to ensure that the # is parsed as a revision specifier, and not as a comment field in the form.

Name	Type	Description
view	list[str]	A list of depot files that can be tagged with this label. No files are actually tagged until `p4 labelsync` is invoked. Unlike client views or branch views, which map one set of files to another, label views consist of a simple list of depot files.
serverID	str	If set, restricts usage of the label to the named server. If unset, this label may be used on any server.

Location

A consolidated mechanism for identifying something that generally has a path in the system.

Each location references either a depot, a dir, or a file.

f .Attributes

Name	Type	Description
depotPath	str	An absolute depot path specification.
depot	[“DepotsCommand” on page 329](#)	
dir	[“DirsCommand” on page 331](#)	
file	[“FilesCommand” on page 331](#)	
fstat	[“FstatCommand” on page 332](#)	
content	str	If this location indicates a single file, this can be set with the Base64-encoded content of the file.

LoginRequest

Captures the login information we need for logging into either a p4d server or our "authentication source".

f .Attributes

Name	Type	Description
user	str	Usually the Perforce username
password	str	
serverLogins	list["ServerLoginRequest" on page](#)	

ServerLoginRequest

f .Attributes

Name	Type	Description
id	str	The server's ID
user	str	
password	str	

LoginResponse

Either of our login methods return a ticket, which is then used as a password in a basic authentication scheme.

When this is returned from the explicit p4d login, this is a host unlocked ticket, acceptable for using with a local client.

f .Attributes

Name	Type	Description
ticket	str	

P4dConfigId

Identification of servers the Helix Web Services instance can connect to.

f .Attributes

Name	Type	Description
id	str	A simple string identifier (alphanumeric characters only, please)
name	str	A display string, not guaranteed to be unique
description	str	A simple textual description, for potential selection by clients.

Protections

Displays the information stored in the `p4 protect` command.

f .Attributes

Name	Type	Description
protections	list[str]	Each item in the protections array is a line in the protections table, and is split into five columns.
		1. Access level or mode. One of the access levels list, read, open, write, admin, super, review; or one of the rights =read, =open, =write, and =branch,
		2. Either **user** or **group**, to indicate what's identified by this entry.
		3. The group name or user name. To grant permission to all users, use a wildcard with just an asterix symbol.
		4. The IP address of the client host.
		5. The depot file path, which can contain wildcards. To exclude this mapping from the permission set, use a dash - as the first character of this value.
		IPv6 addresses and IPv4 addresses are also supported. You can use the * wildcard to refer to all IP addresses, but only when you are not using CIDR notation.
		If you use the * wildcard with an IPv6 address, you must enclose the entire IPv6 address in square brackets. For example,

Name	Type	Description
		[2001:db8:1:2:*] is equivalent to [2001:db8:1:2::]/64. Best practice is to use CIDR notation, surround IPv6 addresses with brackets, and to avoid the * wildcard.
		How the system forms host addresses depends on the setting of the dm.proxy.protects variable. By default, this variable is set to 1. This means that if the client host uses some intermediary (proxy, broker, replica) to access the server, the proxy- prefix is prepended to the client host address to indicate that the connection is not direct. If you specify proxy-* for the Host field, that will affect all connections made via proxies, brokers, and replicas. A value like proxy-10.0.0.5 identifies a client machine with an IP address of 10.0.0.5 that is connected to the server through an intermediary.
		Setting the dm.proxy.protects variable to 0, removes the proxy- prefix and allows you to write a single set of protection entries that apply both to directly-connected clients as well as to those that connect via an intermediary. This is more convenient but less secure if it matters that a connection is made using an intermediary. If you use this setting, all intermediaries must be at release 2012.1 or higher.

ServersCommand

f .Attributes

Name	Type	Description
serverID	str	A unique identifier for this server. This must match the contents of the server's `server.id` file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.
type	str	Server executable type. One of the following: `server`, `proxy`, `broker`, `identifier`, `admin`. Each type may offer one or more services, defined in the `services` property.
services	str	The `server` type server provides the following services: • standard - a standard Perforce server • replica - a read-only replica server • commit-server - central server in distributed installation • edge-server - node in distributed installation • forwarding-replica - a replica configured to forward commands that involve database writes to a master server • build-server - a replica that supports build automation and build farm integration • P4AUTH - a server that provides authentication • P4CHANGE - a server that provides change numbering

Name	Type	Description
		• depot-master - commit-server with automated failover
		• depot-standby - standby replica of the depot-master
		• workspace-server - node in a cluster installation
		• standby - read-only replica server that uses p4 journalcopy
		• forwarding-standby - forwarding replica server that uses p4 journalcopy
		The **proxy** type server provides a p4p caching proxy.
		The **broker** type server provides the following services:
		• broker - a p4broker process
		• workspace-router - routing broker for a cluster
		The services field for the **identifier** type server specifies the existence of the cluster, and has the value **cluster**. The name of the cluster is then drawn from the ServerID field.
		The **admin** type server provides the following services:
		• hxca-server - the admin server for a Helix cluster.
		• zookeeper-server - ZooKeeper server for a cluster
name	str	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.

Name	Type	Description
address	str	The P4PORT used by this server.
description	str	An optional description for this server.
user	str	The service user name used by the server.

ServerCommand

The Perforce server specification describes the high-level configuration and intended usage of a Perforce server. For installations with only one Perforce server, the server specification is optional.

f .Attributes

Name	Type	Description
serverID	str	A unique identifier for this server. This must match the contents of the server's `server.id` file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.
type	str	Server executable type. One of the following: `server`, `proxy`, `broker`, `identifier`, `admin`. Each type may offer one or more services, defined in the `services` property.
services	str	The `server` type server provides the following services: • standard - a standard Perforce server • replica - a read-only replica server • commit-server - central server in distributed installation

Name	Type	Description
		• edge-server - node in distributed installation
		• forwarding-replica - a replica configured to forward commands that involve database writes to a master server
		• build-server - a replica that supports build automation and build farm integration
		• P4AUTH - a server that provides authentication
		• P4CHANGE - a server that provides change numbering
		• depot-master - commit-server with automated failover
		• depot-standby - standby replica of the depot-master
		• workspace-server - node in a cluster installation
		• standby - read-only replica server that uses p4 journalcopy
		• forwarding-standby - forwarding replica server that uses p4 journalcopy
		The **proxy** type server provides a p4p caching proxy.
		The **broker** type server provides the following services:
		• broker - a p4broker process
		• workspace-router - routing broker for a cluster
		The services field for the **identifier** type server specifies the existence of the cluster, and has the value

Name	Type	Description
		`cluster`. The name of the cluster is then drawn from the ServerID field.
		The `admin` type server provides the following services:
		• hxca-server - the admin server for a Helix cluster.
		• zookeeper-server - ZooKeeper server for a cluster
name	str	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.
address	str	The P4PORT used by this server.
externalAddress	str	For an edge server, this optional field specifies the external address used for connections to a commit server. This field must be set for the edge server to enable parallel submits in a federated environment.
description	str	An optional description for this server.
user	str	The service user name used by the server.
clientDataFilter	str	For a replica server, this optional field can contain one or more patterns describing how active client workspace metadata is to be filtered. Active client workspace data includes have lists, working records, and pending resolves.
		To include client data, use the syntax: **//client-pattern/…**

Name	Type	Description
		To exclude client data, use the syntax: `-//client-pattern/…`
		All patterns are specified in client syntax.
revisionDataFilter	str	For a replica server, this optional field can contain one or more patterns describing how submitted revision metadata is to be filtered. Submitted revision data includes revision records, integration records, label contents, and the files listed in submitted changelists.
		To include depot data, use the syntax:
		To exclude depot data, use the syntax: -//depot/pattern/…
		All patterns are specified in depot syntax.
archiveDataFilter	str	For a replica server, this optional field can contain one or more patterns describing the policy for automatically scheduling the replication of file content. If this field is present, only those files described by the pattern are automatically transferred to the replica; other files are not transferred until they are referenced by a replica command that needs the file content.
		Files specified in the ArchiveDataFilter: field are transferred to the replica regardless of whether any users of the replica have made requests for their content.

Name	Type	Description
		To automatically transfer files on submit, use the syntax: `//depot/pattern/…`
		To exclude files from automatic transfer, use the syntax: `-//depot/pattern/…`
		All patterns are specified in depot syntax.
distributedConfig	str	For an edge or commit server, this optional field, which is displayed only when you use the -l or -c option, shows configuration settings for this server.
		`-l` flag shows the current configuration. `-c-` flag shows current configuration values, recommended default values for fields that are not set, or unset for fields that are not set and do not have default values.
		If this field is present when invoked with -c, the configuration commands in this field are run on the current server using the scope of the server specified in the serverID field.

StreamCommand

The Perforce stream specification defines a single stream.

Streams are hierarchical branches with policies that control the structure and the flow of change. Stream hierarchies are based on the stability of the streams, specified by the type you assign to the stream. Development streams are least stable (most subject to change), mainline streams are somewhat stable, and release streams are highly stable. Virtual streams can be used to copy and merge between parent and child streams without storing local data. Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data.

Stream contents are defined by the paths that you map. By default, a stream has the same structure as its parent (the stream from which it was branched), but you can override the structure, for example to ensure that specified files cannot be submitted or integrated to other streams.

f .Attributes

Name	Type	Description
stream	str	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form **//depotname/streamname**.
update	datetime	The date the stream specification was last modified.
access	datetime	The date and time that the stream specification was last accessed by any Perforce command.
owner	str	The Perforce user or group who owns the stream. The default is the user who created the stream.
name	str	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname portion of the stream path.
parent	str	The parent of this stream. Must be none if the stream's **Type:** is mainline, otherwise must be set to an existing stream identifier of the form **//depotname/streamname**.
type	str	The stream's type determines the expected flow of change. Valid stream types are **mainline**, **virtual**, **development**, and **release**. • **mainline**: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream. • **virtual**: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed

Name	Type	Description
		through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream. • `release`: More stable than the mainline. Release streams copy from the parent and merge to the parent. • `development`: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent. • `task`: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.
description	str	Description of the stream.
options	str	Settings that configure stream behavior as follows:

Name	Type	Description
		• `[un]locked`: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification cannot be deleted, and only its owner can modify it. The default is unlocked. • `[all,owner]submit`: Specifies whether all users or only the owner of the stream can submit changes to the stream. The default is allsubmit. If the Owner: of a stream marked ownersubmit is a group, all users who are members of that group can submit changes to the stream. • `[no]toparent`: Specifies whether integrations from the stream to its parent are expected. The default is toparent. • `[no]fromparent`: Specifies whether integrations to the stream from its parent are expected. The default is fromparent for mainline and development streams, and nofromparent for release streams. • `mergeany,mergedown`: Specifies whether the merge flow is restricted or whether merge is permitted from any other stream. For example, the mergeany option would allow a merge from a child to a parent with no warnings. A virtual stream must have its flow options set to notoparent and nofromparent. Flow options are ignored for mainline streams.

Name	Type	Description
paths	list[str]	Paths define how files are incorporated into the stream structure. Specify paths using the following format: `path_type view_path [depot_path]` where `path_type` is a single keyword, `view_path` is a file path with no leading slashes, and the optional `depot_path` is a file path beginning with `//`. The default path is `share` … Valid path types are: • `share view_path`: Specified files can be synced, submitted, and integrated to and from the parent stream. • `isolate view_path`: Specified files can be synced and submitted, but cannot be integrated to and from the parent stream. • `import view_path [depot_path]`: Specified files can be synced, but cannot be submitted or integrated to and from the parent stream. The `view_path` is mapped as in the parent stream's view, or to an (optional) `depot_path`. The `depot_path` may include a changelist specifier. That stream's client workspaces will be limited to seeing revisions at that change or lower within that depot path. For example, you can specify a depot path like this: `//depot/import/…@1000`. Revisions from changelists greater than 1000 will be automatically hidden from most commands. The changelist limits in

Name	Type	Description
		effect for a given stream workspace are displayed in a read-only client workspace specification field called ChangeView.
		• `import+ view_path [depot_path]`: Functions like a standard import path, enabling you to map a path from outside the stream depot to your stream, but unlike a standard import path, you can submit changes to the files in an import+ path.
		• `exclude view_path`: Specified files cannot be synced, submitted or integrated to and from the parent stream. By default, streams inherit their structure from the parent stream (except mainlines, which have no parent). Paths are inherited by child stream views; a child stream's path can downgrade the inherited view, but not upgrade it. (For example, a child stream can downgrade a shared path to an isolated path, but if the parent stream defines a path as isolated, its child cannot restore full access by specifying the path as shared.) Note that the `depot_path` is relevant only when the path_type is `import` or `import+`.
remapped	list[str]	Reassigns the location of workspace files. To specify the source path and its location in the workspace, use the following syntax: `view_path_1 view_path_2` where `view_path_1` and `view_path_2` are Perforce view

Name	Type	Description
		paths (omit leading slashes and leading or embedded wildcards; terminal wildcards are fine). For example, to ensure that files are synced to the local ProjectX folder, remap as follows: … **projectX/…** Line ordering in the Remapped: field is significant: if more than one line remaps the same files, the later line takes precedence. Remappings are inherited by child streams and the workspaces associated with them.
ignored	list[str]	A list of file or directory names to be ignored in client views. For example: ``` /tmp # ignores files named "tmp" /tmp/... # ignores directories named "tmp" .tmp # ignores file names ending in .tmp ``` Lines in the **Ignored:** field can appear in any order. Ignored files and directories are inherited by child stream client views.

StreamsCommand

A summary of a stream in the system, as provided by the **p4 streams** command.

f .Attributes

Name	Type	Description
stream	str	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form **//depotname/streamname**.
update	datetime	The date the stream specification was last modified.

Name	Type	Description
access	datetime	The date and time that the stream specification was last accessed by any Perforce command.
owner	str	The Perforce user or group who owns the stream. The default is the user who created the stream.
name	str	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname portion of the stream path.
parent	str	The parent of this stream. Must be none if the stream's `Type:` is mainline, otherwise must be set to an existing stream identifier of the form `//depotname/streamname`.
type	str	The stream's type determines the expected flow of change. Valid stream types are `mainline`, `virtual`, `development`, and `release`. • `mainline`: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream. • `virtual`: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot

Name	Type	Description
		namespace, it is impossible to import a virtual stream. • `release`: More stable than the mainline. Release streams copy from the parent and merge to the parent. • `development`: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent. • `task`: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.
description	str	Description of the stream.
options	str	Settings that configure stream behavior as follows: • `[un]locked`: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification cannot be deleted, and only its owner can modify it. The default is unlocked.

Name	Type	Description
		• `[all

Triggers

Defines the triggers table, like it would appear in the output to the `p4 triggers` command.

f .Attributes

Name	Type	Description
triggers	list[str]	A list of trigger definitions. A trigger definition contains four fields that specify the name of the trigger, the type of event that should trigger the execution of the script, the paths that should be affected by the trigger, the location of the script, and other trigger type-dependent information. When the condition specified in a trigger definition is satisfied, the associated script or program is executed. Example: `trig1 change-submit //depot/dir/… "/usr/bin/s1.pl %changelist%"` See the Helix Versioning Engine Administrator Guide for more details on trigger definitions.

UserCommand

Create or edit Perforce user specifications and preferences.

There are three types of Perforce users: standard users, operator users, and service users. Standard users are the default, and each standard user consumes one Perforce license. The operator user type is intended for system administrators; they are subject to the same restrictions on permissions as any other user, but are further restricted in that they can run only a limited subset of Perforce commands. Service users are intended for inter-server communication in replicated and multi-server environments, and are restricted to an even smaller subset of Perforce commands. Neither operators nor service users consume Perforce licenses.

f .Attributes

Name	Type	Description
user	str	The Perforce username.
type	str	Type of user: standard, operator, or service. Once you set the type, you cannot change it.
authMethod	str	One of the following: perforce or ldap. Specifying perforce enables authentication using Perforce's internal db.user table or by way of an authentication trigger. This is the default unless it is overridden with the auth.default.method configurable. Specifying ldap enables authentication against AD/LDAP servers specified by the currently active LDAP configurations.
email	str	The user's email address. By default, this is user@client.
update	datetime	The date and time this specification was last updated.
access	datetime	The date and time this user last ran a Perforce command.
fullName	str	The user's full name.
jobView	str	Jobs matching this jobview appear on any changelists created by this user. Jobs that are fixed by the changelist should be left in the changelist when it's submitted with p4 submit; other jobs should be deleted from the form before submission.
password	str	The user's password.

Name	Type	Description
passwordChange	datetime	The date and time of the user's last password change. If the user has no password, this field is blank.
reviews	list[str]	A list of files the user would like to review. This field can include exclusionary mappings.

UsersCommand

f .Attributes

Name	Type	Description
user	str	The Perforce username.
type	str	Type of user: standard, operator, or service. Once you set the type, you cannot change it.
email	str	The user's email address. By default, this is user@client.
update	datetime	The date and time this specification was last updated.
access	datetime	The date and time this user last ran a Perforce command.
fullName	str	The user's full name.
hasPassword	str	If 'enabled', the password has been set on the user.

Ruby SDK Reference

Getting Started

Inside the `clients/ruby` of the installation is an SDK for Ruby 2+ applications. To get started, just run `gem install helix-web-services-client-2016.1.0.gem` and you should have the Ruby client available to your ruby distribution. You may need to install dependencies, which is beyond the scope of this documentation - we assume you are familiar and comfortable developing Ruby applications.

Typically, you'll create an instance of the `DefaultApi` and setup authentication for a particular user.

```
api = HelixWebServices::DefaultApi.new
api.api_client.config.scheme = 'https'
api.api_client.config.host = 'myhost.example.com'

# The default configuration of HWS uses a self-signed certificate, so you will likely
# need to disable SSL verification
api.api_client.config.verify_ssl = false

login_request = HelixWebServices::LoginRequest.new(:user => 'jdoe', :password => 'mypassword')
login_response = api.login_post(login_request)

# We generate an authentication key that is to be used as the Authorization header.
api.api_client.config.api_key['Authorization'] = login_response.ticket

# List depots on p4d 'myserver'
depots = api.servers_depots_get('myserver')
```

HelixWebServices::ApiClient Reference

The `ApiClient` is an attribute on the main API object you instantiate. For example, you will create an instance of `DefaultApi`, and you will access the `ApiClient` via the `api_client` accessor of your `DefaultApi` object.

```
api = HelixWebServices::DefaultApi.new
api.api_client
```

Table 636. Accessors

Name	Type	Description
config	HelixWebServices::Configuration	See "HelixWebServices::Configuration Reference" on page 369.
default_headers	Hash	Adds Request headers to be used in all HTTP requests (by default).

HelixWebServices::Configuration Reference

For the Ruby SDK, the `Configuration` handle is accessed via the `config` accessor of the `ApiClient`, see "HelixWebServices::ApiClient Reference" on page 369.

The Configuration handle is used to create common settings typically needed for most calls to the server.

Table 637. Accessors

Name	Type	Description
api_key	Hash	Defines API keys used with API Key authentication. Use set

Name	Type	Description
		the user specific authentication key via the `Authorization` key on this hash.
base_path	String	Defines URL base path (defaults to `/api/HWS_VERSION`)
cert_file	String	Client certificate file (for client certificate)
debugging	Boolean	Set this to enable/disable debugging. When enabled (set to true), HTTP request/response details will be logged with `logger.debug` (see the `logger` attribute).
host	String	Defines URL host
key_file	String	Client private key file (for client certificate)
scheme	String	Defines URL scheme
ssl_ca_cert	String	Set this to customize the certificate file to verify the peer. See also: The `cainfo` option of Typhoeus, `--cert` option of libcurl.
timeout	Integer	The time limit for HTTP request in seconds. Default to 0 (never times out).
verify_ssl	Boolean	Set this to false to skip verifying SSL certificate when calling API from https server. Default to true.

HelixWebServices::DefaultApi Reference

DefaultApi#config_p4ds_get

Method Signature.

```
Array HelixWebServices::DefaultApi#config_p4ds_get()
```

The list of registered p4d servers in your cluster.

This is provided by a special set of configuration files in the system. For more information, consult the Helix Web Services user guide.

Table 638. Returns

Type	Notes
Array of ["P4dConfigId" on page 445](#)	

DefaultApi#login_post

Method Signature.

```
LoginResponse HelixWebServices::DefaultApi#login_post(loginRequest)
```

Logs into the primary authentication source.

This can either be a p4d instance or Helix Cloud, depending upon the configuration of your Helix Web Services instance.

Table 639. Parameters

Name	Type	Description	Required
loginRequest	LoginRequest	The user login and password.	true

Table 640. Returns

Type	Notes
["LoginResponse" on page 444](#)	Object with ticket to use for Basic auth password.

DefaultApi#status_get

Method Signature.

```
HWSStatus HelixWebServices::DefaultApi#status_get()
```

A simple structure to monitor for "problems" an admin should take care of, and, report the current application version.

This method does not require authentication.

Table 641. Returns

Type	Notes
["HWSStatus" on page 440](#)	

DefaultApi#server_branches_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_branches_get(server)
```

Lists available branches in the system. The resources of this list are summaries of branches in the system.

Table 642. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 643. Returns

Type	Notes
Array of "BranchesCommand" on page 40	Summaries of branches in the system.

DefaultApi#server_branches_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_branches_post(server, body)
```

Creates a new branch specification, like the `p4 branch` command.

Table 644. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	BranchCommand	The branch specification.	true

Table 645. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_branches_branch_get

Method Signature.

```
BranchCommand HelixWebServices::DefaultApi#server_branches_branch_get(server, branch)
```

Returns the branch spec details of the particular branch.

Table 646. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
branch	String	The branch ID	true

Table 647. Returns

Type	Notes
"BranchCommand" on page 406	Branch spec details

DefaultApi#server_branches_branch_patch

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_branches_branch_patch(server, branch, body)
```

Update branch specifications, similar to the `p4 branch` command. Only the specified parameters in the body will be changed.

Table 648. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
branch	String	The branch ID	true
body	BranchCommand	Fields of the branch to update	true

Table 649. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_branches_branch_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_branches_branch_delete(server, branch)
```

Removes the branch specification, similar to the `p4 branch -d` command.

Table 650. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
branch	String	The branch ID	true

Table 651. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_changes_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_changes_get(server, opts)
```

Lists available changes in the system. The resources of this list are summaries of changes in the system.

Table 652. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
opts	Hash	One of the following values: • `max`: Limit the number of change results • `status`: The status of the changes, e.g., `submitted` • `user`: The user's login who submitted the change • `files`: Limit changes to the depot path expressions. See the changes command description.	false

Table 653. Returns

Type	Notes
Array of "ChangesCommand" on page 41(	Summaries of changes in the system.

DefaultApi#server_changes_change_get

Method Signature.

```
ChangeCommand HelixWebServices::DefaultApi#server_changes_change_get(server, change)
```

Returns the change spec details of the particular change.

Table 654. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
change	String	The change ID	true

Table 655. Returns

Type	Notes
"ChangeCommand" on page 408	Change spec details

DefaultApi#server_clients_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_clients_get(server)
```

Lists available clients in the system. The resources of this list are summaries of clients in the system.

Table 656. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 657. Returns

Type	Notes
Array of "ClientsCommand" on page 418	Summaries of clients in the system.

DefaultApi#server_clients_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_clients_post(server, client)
```

Creates a new client specification, like the `p4 client` command.

Table 658. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
client	ClientCommand	The client spec	true

Table 659. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_clients_client_get

Method Signature.

```
ClientCommand HelixWebServices::DefaultApi#server_clients_client_get(server, client)
```

Returns the client spec details of the particular client.

Table 660. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
client	String	The client ID	true

Table 661. Returns

Type	Notes
"ClientCommand" on page 412	Client spec details

DefaultApi#server_clients_client_patch

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_clients_client_patch(server, client, body)
```

Update client specifications, similar to the `p4 client` command. Only the specified parameters in the body will be changed.

Table 662. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
client	String	The client ID	true
body	ClientCommand	Fields of the client to update	true

Table 663. Returns

Type	Notes
[“CommandResponse” on page 42](#)	

DefaultApi#server_clients_client_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_clients_client_delete(server, client)
```

Removes the client specification, similar to the `p4 client -d` command.

Table 664. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
client	String	The client ID	true

Table 665. Returns

Type	Notes
[“CommandResponse” on page 42](#)	

DefaultApi#server_commands_command_get

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_commands_command_get(server, command, opts)
```

Execute a Perforce command that requires no input. This only allows commands that have been whitelisted on your system. See the [“Configuration” on page 23](#) section for details.

Table 666. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
command	String	The command name	true
opts	Hash	One of the following values: • **arg**: Command arguments	false

Table 667. Returns

Type	Notes
"CommandResponse" on page 42	Generic list of hashes response

DefaultApi#server_commands_command_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_commands_command_post(server, command, input,
 opts)
```

Execute a Perforce command that accepts input, like a spec. This only allows commands that have been whitelisted on your system. See the "Configuration" on page 23 section for details.

Table 668. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
command	String	The command name	true
input	CommandRequest	A hash used as input to the command	false
opts	Hash	One of the following values: • **arg**: Command arguments	false

Table 669. Returns

Type	Notes
"CommandResponse" on page 42	Generic list of hashes response

DefaultApi#server_counters_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_counters_get(server)
```

Lists available counters in the system. The resources of this list are summaries of counters in the system.

Table 670. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 671. Returns

Type	Notes
Array of "Counter" on page 422	Summaries of counters in the system.

DefaultApi#server_counters_counter_put

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_counters_counter_put(server, counter, body)
```

Update counter specifications, similar to the `p4 counter` command. Only the specified parameters in the body will be changed.

Table 672. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
counter	String	The counter ID	true
body	Counter	Fields of the counter to update	true

Table 673. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_counters_counter_get

Method Signature.

```
Counter HelixWebServices::DefaultApi#server_counters_counter_get(server, counter)
```

Returns the counter spec details of the particular counter.

Table 674. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
counter	String	The counter ID	true

Table 675. Returns

Type	Notes
"Counter" on page 422	Counter spec details

DefaultApi#server_counters_counter_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_counters_counter_delete(server, counter)
```

Removes the counter specification, similar to the **p4 counter -d** command.

Table 676. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
counter	String	The counter ID	true

Table 677. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_counters_counter_increment_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_counters_counter_increment_post(server,
 counter)
```

Increments a numerical counter, similar to the `p4 counter -i` command.

Table 678. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
counter	String	The counter ID	true

Table 679. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_depots_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_depots_get(server)
```

Lists available depots in the system. The resources of this list are summaries of depots in the system.

Table 680. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 681. Returns

Type	Notes
Array of "DepotsCommand" on page 425	Summaries of depots in the system.

DefaultApi#server_depots_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_depots_post(server, depot)
```

Creates a new depot specification, like the `p4 depot` command.

Table 682. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
depot	DepotCommand	The depot spec	true

Table 683. Returns

Type	Notes

DefaultApi#server_depots_depot_get

Method Signature.

```
DepotCommand HelixWebServices::DefaultApi#server_depots_depot_get(server, depot)
```

Returns the depot spec details of the particular depot.

Table 684. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
depot	String	The depot ID	true

Table 685. Returns

Type	Notes
	Depot spec details

DefaultApi#server_depots_depot_patch

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_depots_depot_patch(server, depot, body)
```

Update depot specifications, similar to the `p4 depot` command. Only the specified parameters in the body will be changed.

Table 686. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
depot	String	The depot ID	true
body	DepotCommand	Fields of the depot to update	true

Table 687. Returns

Type	Notes
[“CommandResponse” on page 42](#)	

DefaultApi#server_depots_depot_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_depots_depot_delete(server, depot)
```

Removes the depot specification, similar to the `p4 depot -d` command.

Table 688. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
depot	String	The depot ID	true

Table 689. Returns

Type	Notes
[“CommandResponse” on page 42](#)	

DefaultApi#server_groups_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_groups_get(server)
```

Lists available groups in the system. The resources of this list are summaries of groups in the system.

Table 690. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 691. Returns

Type	Notes
Array of "GroupsCommand" on page 439	Summaries of groups in the system.

DefaultApi#server_groups_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_groups_post(server, body)
```

Creates a new group specification, like the **p4 group** command.

Table 692. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	GroupCommand	The group spec	true

Table 693. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_groups_group_get

Method Signature.

```
GroupCommand HelixWebServices::DefaultApi#server_groups_group_get(server, group)
```

Returns the group spec details of the particular group.

Table 694. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
group	String	The group ID	true

Table 695. Returns

Type	Notes
"GroupCommand" on page 437	Group spec details

DefaultApi#server_groups_group_patch

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_groups_group_patch(server, group, body)
```

Update group specifications, similar to the `p4 group` command. Only the specified parameters in the body will be changed.

Table 696. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
group	String	The group ID	true
body	GroupCommand	Fields of the group to update	true

Table 697. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_groups_group_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_groups_group_delete(server, group)
```

Removes the group specification, similar to the `p4 group -d` command.

Table 698. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
group	String	The group ID	true

Table 699. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_jobs_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_jobs_get(server)
```

Lists available jobs in the system. The resources of this list are summaries of jobs in the system.

Table 700. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 701. Returns

Type	Notes
Array of "JobsCommand" on page 440	Summaries of jobs in the system.

DefaultApi#server_jobs_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_jobs_post(server, job)
```

Creates a new job specification, like the `p4 job` command.

Table 702. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	JobCommand	The job spec	true

Table 703. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_jobs_job_get

Method Signature.

```
JobCommand HelixWebServices::DefaultApi#server_jobs_job_get(server, job)
```

Returns the job spec details of the particular job.

Table 704. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	String	The job ID	true

Table 705. Returns

Type	Notes
"JobCommand" on page 440	Job spec details

DefaultApi#server_jobs_job_patch

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_jobs_job_patch(server, job, jobCommand)
```

Update job specifications, similar to the **p4 job** command. Only the specified parameters in the body will be changed.

Table 706. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	String	The job ID	true
jobCommand	JobCommand	Fields of the job to update	true

Table 707. Returns

Type	Notes
[“CommandResponse” on page 42](#)	

DefaultApi#server_jobs_job_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_jobs_job_delete(server, job)
```

Removes the job specification, similar to the `p4 job -d` command.

Table 708. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	String	The job ID	true

Table 709. Returns

Type	Notes
[“CommandResponse” on page 42](#)	

DefaultApi#server_jobs_job_fixes_change_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_jobs_job_fixes_change_delete(server, job,
  change)
```

Removes the fix record association for the job for a particular changelist.

Table 710. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	String	The job ID	true
change	String	The change ID	true

Table 711. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_jobs_job_fixes_change_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_jobs_job_fixes_change_post(server, job,
  change, opts)
```

Adds a fix record to the job for a particular changelist.

Table 712. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
job	String	The job ID	true
change	String	The change ID	true
opts	Hash	One of the following values: • `status`: Specify the job status instead of using the default. The default is typically `closed` or some other value defined in the `Presets` field specified in the `p4 jobspec` form. If the changelist to which you're linking the job been submitted, the status value is immediately reflected in the job's status.	false

Name	Type	Description	Required
		If the changelist is pending, the job status is changed on submission of the changelist, provided that the -s option is also supplied to p4 submit and the desired status appears next to the job in the p4 submit form's Jobs: field. To leave a job unchanged, use the special status of same.	

Table 713. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_labels_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_labels_get(server)
```

Lists available labels in the system. The resources of this list are summaries of labels in the system.

Table 714. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 715. Returns

Type	Notes
Array of "LabelsCommand" on page 440	Summaries of labels in the system.

DefaultApi#server_labels_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_labels_post(server, label)
```

Creates a new label specification, like the `p4 label` command.

Table 716. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
label	LabelCommand	The label spec	true

Table 717. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_labels_label_get

Method Signature.

```
LabelCommand HelixWebServices::DefaultApi#server_labels_label_get(server, label)
```

Returns the label spec details of the particular label.

Table 718. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
label	String	The label ID	true

Table 719. Returns

Type	Notes
"LabelCommand" on page 442	Label spec details

DefaultApi#server_labels_label_patch

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_labels_label_patch(server, label,
  labelCommand)
```

Update label specifications, similar to the `p4 label` command. Only the specified parameters in the body will be changed.

Table 720. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
label	String	The label ID	true
labelCommand	LabelCommand	Fields of the label to update	true

Table 721. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_labels_label_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_labels_label_delete(server, label)
```

Removes the label specification, similar to the `p4 label -d` command.

Table 722. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
label	String	The label ID	true

Table 723. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_login_post

Method Signature.

```
LoginResponse HelixWebServices::DefaultApi#server_login_post(server, body)
```

Logs into a Helix Versioning Engine (p4d) server.

Table 724. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	LoginRequest	The user login and password.	true

Table 725. Returns

Type	Notes
"LoginResponse" on page 444	Object with ticket to use for Basic auth password.

DefaultApi#server_paths_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_paths_get(server, opts)
```

Lists depots, files, and directories in the system. This combines the output of the `p4 depots`, `p4 dirs`, and `p4 files` commands, depending upon your path.

Table 726. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
opts	Hash	One of the following values: • `path`: The path "under a depot" to query under, e.g., `//depot/main`. This will list the directories and files underneath that path.	false

Table 727. Returns

Type	Notes
Array of "Location" on page 443	Array of depots.

DefaultApi#server_protections_put

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_protections_put(server, protections)
```

Updates the protections table.

This method requires superuser access.

Table 728. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
protections	Protections	The new protections table	true

Table 729. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_protections_get

Method Signature.

```
Protections HelixWebServices::DefaultApi#server_protections_get(server)
```

Returns a list of available protections in the system. The elements of this list are rows of the system's protections table.

This method requires superuser access.

See the output of **p4 protect** for more information.

Table 730. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 731. Returns

Type	Notes
"Protections" on page 445	Object including list of protections entries

DefaultApi#server_servers_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_servers_get(server)
```

Lists available servers in the system. The resources of this list are summaries of servers in the system.

Table 732. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 733. Returns

Type	Notes
Array of "ServersCommand" on page 447	Summaries of servers in the system.

DefaultApi#server_servers_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_servers_post(server, serverCommand)
```

Creates a new server specification, like the `p4 server` command.

Table 734. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
serverCommand	ServerCommand	The server spec	true

Table 735. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_servers_serverid_get

Method Signature.

```
ServerCommand HelixWebServices::DefaultApi#server_servers_serverid_get(server, serverId)
```

Returns the server spec details of the particular server.

Table 736. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
serverId	String	The server ID of the server spec	true

Table 737. Returns

Type	Notes
["ServerCommand" on page 449](#)	Server spec details

DefaultApi#server_servers_serverid_patch

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_servers_serverid_patch(server, serverId,
  serverCommand)
```

Update server specifications, similar to the `p4 server` command. Only the specified parameters in the body will be changed.

Table 738. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
serverId	String	The server ID	true
serverCommand	ServerCommand	Fields of the server to update	true

Table 739. Returns

Type	Notes
["CommandResponse" on page 42](#)	

DefaultApi#server_servers_serverid_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_servers_serverid_delete(server, serverId)
```

Removes the server specification, similar to the `p4 server -d` command.

Table 740. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
serverId	String	The server ID	true

Table 741. Returns

Type	Notes
<u>"CommandResponse" on page 42</u>	

DefaultApi#server_streams_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_streams_get(server)
```

Lists available streams in the system. The resources of this list are summaries of streams in the system.

Table 742. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 743. Returns

Type	Notes
Array of <u>"StreamsCommand" on page 460</u>	Summaries of streams in the system.

DefaultApi#server_streams_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_streams_post(server, body)
```

Creates a new stream specification, like the `p4 stream` command.

Table 744. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	StreamCommand	The stream spec	true

Table 745. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_streams_stream_get

Method Signature.

```
StreamCommand HelixWebServices::DefaultApi#server_streams_stream_get(server, opts)
```

Returns the stream spec details of the particular stream.

Table 746. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
opts	Hash	One of the following values: • **stream**: The stream ID	false

Table 747. Returns

Type	Notes
"StreamCommand" on page 454	Stream spec details

DefaultApi#server_streams_stream_patch

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_streams_stream_patch(server, body, opts)
```

Update stream specifications, similar to the `p4 stream` command. Only the specified parameters in the body will be changed.

Table 748. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	StreamCommand	Fields of the stream to update	true
opts	Hash	One of the following values: • `stream`: The stream ID	false

Table 749. Returns

Type	Notes
[“CommandResponse” on page 42](#)	

DefaultApi#server_streams_stream_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_streams_stream_delete(server, opts)
```

Removes the stream specification, similar to the `p4 stream -d` command.

Table 750. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
opts	Hash	One of the following values: • `stream`: The stream ID	false

Table 751. Returns

Type	Notes
[“CommandResponse” on page 42](#)	

DefaultApi#server_triggers_put

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_triggers_put(server, triggers)
```

Updates the triggers table.

This method requires superuser access.

Table 752. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
triggers	Triggers	The new triggers table	true

Table 753. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_triggers_get

Method Signature.

```
Triggers HelixWebServices::DefaultApi#server_triggers_get(server)
```

Returns a list of available triggers in the system. The elements of this list are rows of the system's triggers table.

This method requires superuser access.

Table 754. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 755. Returns

Type	Notes
"Triggers" on page 462	List of triggers entries

DefaultApi#server_users_get

Method Signature.

```
Array HelixWebServices::DefaultApi#server_users_get(server, opts)
```

Lists available users in the system. The resources of this list are summaries of users in the system.

Table 756. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
opts	Hash	One of the following values: • **includeService**: If true, shows service users in the list. • **max**: Cap the number of users reported to this amount.	false

Table 757. Returns

Type	Notes
Array of <ins>"UsersCommand" on page 464</ins>	Summaries of users in the system.

DefaultApi#server_users_post

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_users_post(server, body)
```

Creates a new user specification, like the `p4 user` command.

Table 758. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	UserCommand	The user spec	true

Table 759. Returns

Type	Notes
<ins>"CommandResponse" on page 42</ins>	

DefaultApi#server_users_user_get

Method Signature.

```
UserCommand HelixWebServices::DefaultApi#server_users_user_get(server, user)
```

Returns the user spec details of the particular user.

Table 760. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
user	String	The user ID	true

Table 761. Returns

Type	Notes
"UserCommand" on page 463	User spec details

DefaultApi#server_users_user_patch

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_users_user_patch(server, user, body)
```

Update user specifications, similar to the `p4 user` command. Only the specified parameters in the body will be changed.

Table 762. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
user	String	The user ID	true
body	UserCommand	Fields of the user to update	true

Table 763. Returns

Type	Notes
"CommandResponse" on page 42	

DefaultApi#server_users_user_delete

Method Signature.

```
CommandResponse HelixWebServices::DefaultApi#server_users_user_delete(server, user)
```

Removes the user specification, similar to the `p4 user -d` command.

Table 764. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
user	String	The user ID	true

Table 765. Returns

Type	Notes
[“CommandResponse” on page 42](#)	

HelixWebServices::AlphaApi Reference

AlphaApi#server_changes_post

Method Signature.

```
CommandResponse HelixWebServices::AlphaApi#server_changes_post(server, changelistRequest)
```

Create a new changelist that can affect multiple files using different kinds of actions. If you require the ability to integrate or move, for example, you can use this method.

Table 766. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
changelistRequest	ChangelistRequest	Description of changes to make	true

Table 767. Returns

Type	Notes
[“CommandResponse” on page 42](#)	

AlphaApi#server_git_fusion_repos_get

Method Signature.

```
Array HelixWebServices::AlphaApi#server_git_fusion_repos_get(server)
```

Lists all configured repositories readable by the current user. .Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true

Table 768. Returns

Type	Notes
Array of ["GitFusionRepoId" on page 430](#)	List of configured repository names and IDs

AlphaApi#server_git_fusion_repos_post

Method Signature.

```
CommandResponse HelixWebServices::AlphaApi#server_git_fusion_repos_post(server, body)
```

Submits a [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file to create or update a repository configuration.

If the repository does not exist or has been previously deleted, this method saves contents of the config file to a new **p4gf_config** file. If the repository has already been initialised, this method replaces all of the file contents of the specified repository's **p4gf_config** file.

Table 769. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
body	GitFusionRepoCon:	The new configuration	true

Table 770. Returns

Type	Notes
["CommandResponse" on page 42](#)	

AlphaApi#server_git_fusion_repos_repo_get

Method Signature.

```
GitFusionRepoConfig HelixWebServices::AlphaApi#server_git_fusion_repos_repo_get(server, repo)
```

Return configuration for the specified repository. Grabs and returns contents of the [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file for given repository.

Table 771. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
repo	String	The Git Fusion Repo ID	true

Table 772. Returns

Type	Notes
"GitFusionRepoConfig" on page	Git Fusion repository config

AlphaApi#server_git_fusion_repos_repo_patch

Method Signature.

```
CommandResponse HelixWebServices::AlphaApi#server_git_fusion_repos_repo_patch(server, repo, body)
```

Updates values in the repository configuration. This method will find all specified parameters and update each value for the specified repository's configuration file.

Table 773. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
repo	String	The Git Fusion Repo ID	true
body	GitFusionRepoCon:	The new configuration	true

Table 774. Returns

Type	Notes
"CommandResponse" on page 42	

AlphaApi#server_git_fusion_repos_repo_delete

Method Signature.

```
CommandResponse HelixWebServices::AlphaApi#server_git_fusion_repos_repo_delete(server, repo)
```

Deletes the repository configuration (The [p4gf_config file](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j)). Contents of the repository are not deleted from Perforce.

Table 775. Parameters

Name	Type	Description	Required
server	String	The server ID that we execute this particular method against.	true
repo	String	The Git Fusion Repo ID	true

Table 776. Returns

Type	Notes
"CommandResponse" on page 42	

Ruby Models

BranchCommand

Models the output of a `p4 branch` command.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_branch.html).

Table 777. Attributes

Name	Type	Description
branch	String	The branch name, as provided on the command line.
owner	String	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.
access	Date	The date the branch mapping was last accessed.
update	Date	The date the branch mapping was last changed.
options	String	Either `unlocked` (the default) or `locked`. If `locked`, only the `Owner:` can modify the branch mapping,

Name	Type	Description
		and the mapping can't be deleted until it is **unlocked**.
description	String	A short description of the branch's purpose.

view	Array of String	A set of mappings from one set of files in the depot (the source files) to another set of files in the depot (the target files). The view maps from one location in the depot to another; it can't refer to a client workspace.
		
		`For example, the branch view`
		` //depot/main/... //depot/`
		`r2.1/...  maps all the files`
		` under  //depot/main  to //`
		`depot/r2.1 .`

BranchesCommand

A reference to a branch mapping known to the system.

Table 778. Attributes

Name	Type	Description
branch	String	The branch name, as provided on the command line.
owner	String	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.
access	Date	The date the branch mapping was last accessed.
update	Date	The date the branch mapping was last changed.
options	String	Either **unlocked** (the default) or **locked**. If **locked**, only the **Owner:** can modify the branch mapping,

Name	Type	Description
		and the mapping can't be deleted until it is **unlocked**.
description	String	A short description of the branch's purpose.

ChangeCommand

A changelist specification.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_change.html).

Table 779. Attributes

Name	Type	Description
change	String	Contains the changelist number if editing an existing changelist, or **new** if creating a new changelist.
client	String	Name of current client workspace
date	Date	Date the changelist was last modified.
user	String	Name of the change owner. The owner of an empty pending changelist (that is, a pending changelist without any files in it) can transfer ownership of the changelist to another existing user either by editing this field, or by using the -U user option. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
status	String	**pending**, **shelved**, **submitted**, or **new**. Not editable by the user.

Name	Type	Description
		The status is `new` when the changelist is created, `pending` when it has been created but has not yet been submitted to the depot, `shelved` when its contents are shelved, and `submitted` when its contents have been stored in the depot.
description	String	Textual description of changelist. If you do not have access to a restricted changelist, the description is replaced with a "no permission" message.
jobs	Array of String	A list of jobs that are fixed by this changelist.
type	String	Type of change: `restricted` or `public`. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change. Public changes are displayed without restrictions. By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.
files	Array of String	The list of files submitted in this changelist.
importedBy	String	Displays the name of the user who ran the p4 fetch, p4 push, or p4 unzip command that

Name	Type	Description
		imported this change into the server.
		This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied.
		In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.
identify	String	Contains a label which uniquely identifies this changelist across all servers where it has been fetched, pushed, or unzipped.
		This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied.
		In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.

ChangesCommand

Table 780. Attributes

Name	Type	Description
change	String	The changelist ID
date	Date	Last modification time of the changelist

Name	Type	Description
user	String	The owner of the changelist
client	String	Name of current client workspace.
status	String	`pending`, `shelved`, `submitted`, or `new`. Not editable by the user. The status is `new` when the changelist is created, `pending` when it has been created but has not yet been submitted to the depot, `shelved` when its contents are shelved, and `submitted` when its contents have been stored in the depot.
type	String	Type of change: `restricted` or `public`. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change. Public changes are displayed without restrictions. By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.
path	String	Depot paths affected by this changelist
description	String	Textual description of changelist. If you do not have access to a restricted changelist, the

Name	Type	Description
		description is replaced with a "no permission" message.

ChangelistRequest

Table 781. Attributes

Name	Type	Description
description	String	
stream	String	Optional stream ID to use in case you want to edit files in a stream.
actions	Array of "ChangelistAction" on page 412	

ChangelistAction

Table 782. Attributes

Name	Type	Description
depotFile	String	The target file path to edit.
fromDepotFile	String	For "branch" or "move" actions, this indicates the source file location.
actionType	String	One of "upload", "branch", "move", or "delete"
content	String	Base64-encoded content
requireVersion	Number	If set, we will only operate if this is the current version of the file.

ClientCommand

The client workspace specification and its view.

For more information see the [command reference](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html).

Table 783. Attributes

Name	Type	Description
client	String	The client workspace name, as specified in the **P4CLIENT** environment variable or its equivalents.
owner	String	The name of the user who owns the workspace. The default is the user who created the workspace.
		The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
update	Date	The time the workspace specification was last modified.
access	Date	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)
host	String	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host.
		The hostname must be provided exactly as it appears in the output of p4 info when run from that host.
		This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option

Name	Type	Description
		to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.
description	String	A textual description of the workspace. The default text is Created by owner.
root	String	The directory (on the local host) relative to which all the files in the `View:` are specified. The default is the current working directory. The path must be specified in local file system syntax. If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to map files to multiple drives.
altRoots	Array of String	Up to two optional alternate client workspace roots. Perforce applications use the first of the main and alternate roots that match the application's current working directory. Use the p4 info command to display the root being used. This enables users to use the same Perforce client workspace specification on multiple platforms, even those with different directory naming conventions. If you are using multiple or alternate workspace roots (the AltRoots: field), you can always tell which root is in effect by

Name	Type	Description
		looking at the Client root: reported by p4 info.
		If you are using a Windows directory in any of your workspace roots, you must specify the Windows directory as your main workspace root and specify your other workspace roots in the AltRoots: field.
options	String	A set of seven switches that control particular workspace options.
		See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
submitOptions	String	Options to govern the default behavior of p4 submit.
		• submitunchanged
		All open files (with or without changes) are submitted to the depot. This is the default behavior of Perforce.
		• submitunchanged+reopen
		All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist.
		• revertunchanged
		Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted.
		• revertunchanged+reopen

Name	Type	Description
		Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist. • leaveunchanged Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist. • leaveunchanged+reopen Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged+reopen, except that no unchanged files are submitted to the depot.
lineEnd	String	Configure carriage-return/linefeed (CR/LF) conversion. See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options.
stream	String	Associates the workspace with the specified stream. Perforce generates the view for stream-associated workspaces: you cannot modify it manually.

Name	Type	Description
streamAtChange	String	A changelist number that sets a back-in-time view of a stream.
		When StreamAtChange is set, running p4 sync (when called with no arguments) updates the workspace to files at this changelist revision, instead of the head revision. You cannot submit changes (p4 submit returns an error) when StreamAtChange is set, because the workspace view no longer reflects the current stream inheritance.
		This field is ignored unless the Stream field is also set to a valid stream.
serverID	String	If set, restricts usage of the workspace to the named server. If unset, use is allowed on master server and on any replicas of the master other than Edge servers.
view	Array of String	Specifies the mappings between files in the depot and files in the workspace. A new view takes effect on the next p4 sync operation.
changeView	Array of String	Restricts access to depot paths to a particular point in time. Files specified for the ChangeView field are read-only: they may be opened but not submitted. For example: `//depot/path/…@1000`
		Revisions of the files in the specified path will not be visible if they were submitted after the specified changelist number. Files matching a ChangeView path may not be submitted.

Name	Type	Description
type	String	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts. Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.

ClientsCommand

Table 784. Attributes

Name	Type	Description
client	String	The client workspace name, as specified in the P4CLIENT environment variable or its equivalents.
owner	String	The name of the user who owns the workspace. The default is the user who created the workspace. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.

Name	Type	Description
update	Date	The time the workspace specification was last modified.
access	Date	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)
host	String	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host. The hostname must be provided exactly as it appears in the output of p4 info when run from that host. This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.
description	String	A textual description of the workspace. The default text is Created by owner.
root	String	The directory (on the local host) relative to which all the files in the **View:** are specified. The default is the current working directory. The path must be specified in local file system syntax.

Name	Type	Description
		If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to map files to multiple drives. additionalProperties:
type	String	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts.

Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable. |
| options | String | A set of seven switches that control particular workspace options.

See [Usage Notes](https://www.perforce.com/perforce/doc.current/manuals/cmdref/p4_client.html#p4_client.usage) for a listing of these options. |
| submitOptions | String | Options to govern the default behavior of p4 submit.

• submitunchanged

  All open files (with or without changes) are submitted to the depot. This |

Name	Type	Description
		is the default behavior of Perforce.
		• submitunchanged+reopen
		All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist.
		• revertunchanged
		Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted.
		• revertunchanged+reopen
		Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist.
		• leaveunchanged
		Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist.
		• leaveunchanged+reopen
		Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged+reopen, except that no unchanged

Name	Type	Description
		files are submitted to the depot.
lineEnd	String	Configure carriage-return/ linefeed (CR/LF) conversion. See [Usage Notes](https:// www.perforce.com/perforce/ doc.current/manuals/cmdref/ p4_client.html#p4_client.usage) for a listing of these options.
stream	String	Associates the workspace with the specified stream. Perforce generates the view for stream-associated workspaces: you cannot modify it manually.

CommandResponse

A generic container for responses from the p4d server that we have yet to completely classify.

Table 785. Attributes

Name	Type	Description
results	Array of object	A collection of maps that have various values set by p4d.

CommandRequest

A single map typically defines input to generic command methods.

Table 786. Attributes

Name	Type	Description
object	Object	Don't use this. It's a kludge around a bug in the Java client code generator

Counter

A persistent variable in the server.

Table 787. Attributes

Name	Type	Description
counter	String	The variable name
value	String	The variable value. Many variables are numerical in nature, which allow you to do atomic increment operations in method calls instead of having to fetch, increment, and save.

DepotCommand

The depot specification, which is the shared repository Perforce stores files in.

Table 788. Attributes

Name	Type	Description
depot	String	The depot name.
owner	String	The user who owns the depot. By default, this is the user who created the depot. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
description	String	A short description of the depot's purpose. Optional.
type	String	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`. A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server. A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy.

Name	Type	Description
		A remote depot references files that reside on other servers, and cannot be written to.
		The spec depot, if present, automatically archives edited forms.
		The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.
		An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries.
		A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.
address	String	If the `Type:` is `remote`, the address should be the P4PORT address of the remote server. If the `Type:` is local or spec, this field is ignored.
suffix	String	If the Type: is spec, this field holds an optional suffix for generated paths to objects in the spec depot.
		The default suffix is .p4s. You do not need a suffix to use the spec depot, but supplying a file extension to your Perforce server's versioned specs enables users of GUI client software to associate Perforce specifications with a preferred

Name	Type	Description
		text editor. If the Type: is local or remote, this field is ignored.
streamDepth	String	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following the depot name of a stream.] This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.
map	String	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the ... wildcard; for example, a local depot new might have a Map: of `new/...`. If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, `//depot/new/rel2/...` This directory will be the root of the local representation of the remote depot.
specMap	Array of String	For spec depots, an optional description of which specs should be saved, expressed as a view.

DepotsCommand

A summary of depots in the system, with information provided by the `p4 depots` command.

Table 789. Attributes

Name	Type	Description
depot	String	The depot name.

Name	Type	Description
map	String	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the … wildcard; for example, a local depot new might have a Map: of `new/…`. If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, `//depot/new/rel2/…`. This directory will be the root of the local representation of the remote depot.
type	String	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`. A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server. A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy. A remote depot references files that reside on other servers, and cannot be written to. The spec depot, if present, automatically archives edited forms. The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command. An archive depot is used in conjunction with the p4 archive and p4 restore commands

Name	Type	Description
		to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries. A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.
streamDepth	String	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following the depot name of a stream.] This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.
description	String	A short description of the depot's purpose. Optional.

DirsCommand

Table 790. Attributes

Name	Type	Description
dir	String	

FilesCommand

Table 791. Attributes

Name	Type	Description
depotFile	String	
revision	String	
change	String	

Name	Type	Description
action	String	
time	Date	
type	String	

FstatCommand

Detailed information about each file, as provided by the `p4 fstat` command.

Table 792. Attributes

Name	Type	Description
depotFile	String	Depot path to file. For files containing special characters, the filename is displayed containing the ASCII expression of the character's hexadecimal value.
movedFile	String	Name in depot of moved to/from file.
shelved	String	Set to **shelved** if file is shelved.
headAction	String	Action taken at head revision, if in depot. One of: add, edit, delete, branch, move/add, move/delete, integrate, import, purge, or archive.
headChange	String	Head revision changelist number, if in depot.
headRev	String	Head revision number, if in depot.
headType	String	Head revision type, if in depot.
headCharset	String	Head charset, for unicode files.
headTime	Date	Head revision changelist time, if in depot. Time is measured in seconds since 00:00:00 UTC, January 1, 1970.

Name	Type	Description
headModTime	Date	Head revision modification time (the time that the file was last modified on the client before submit), if in depot.
movedRev	String	Head revision of moved file.
digest	String	MD5 digest of a file.
fileSize	String	File length in bytes.
actionOwner	String	User who opened the file, if open.
resolved	String	The number, if any, of resovled integration records.
unresolved	String	The number, if any, of unresolved integration records.
reresolvable	String	The number, if any, of re-resolvable integration records.
otherOpens	Array of String	For each user with the file open, the workspace and user with the open file.
otherLocks	Array of String	For each user with the file locked, the workspace and user holding the lock.
otherActions	Array of String	For each user with the file open, the action taken.
otherChanges	Array of String	The changelist number with this file open.
resolveActions	Array of String	Pending integration action.
resolveBaseFiles	Array of String	Pending base files.
resolveBaseRevs	Array of String	Pending base revision numbers.
resolveFromFiles	Array of String	Pending from files.
resolveStartFromRevs	Array of String	Pending starting revisions.
resolveEndFromRevs	Array of String	Pending ending revisions.

GitFusionRepoId

Table 793. Attributes

Name	Type	Description
id	String	An identifier for the repository that can be used safely within URL paths.
name	String	The repository name, which can be path-like.

GitFusionRepoConfig

Table 794. Attributes

Name	Type	Description
name	String	The repository name, which can be path-like.
description	String	Repo description returned by the @list command.
globalOverrides	"GitFusionRepoGlobalOverrides"	
branches	Array of "GitFusionRepoBranchConfig" o	

GitFusionRepoBranchConfig

Defines a unique Git Fusion branch.

Table 795. Attributes

Name	Type	Description
gitBranchId	String	Alphanumeric ID for the git branch. *Do not change this value once this repo has been cloned*.
gitBranchName	String	Defines a name specified in a local repo for a Git branch. A valid Git branch name. Do not edit this value after you clone the repo.
view	Array of String	Defines a Perforce workspace view mapping that maps

Name	Type	Description
		Perforce depot paths (left side) to Git work tree paths (right side).
		Correctly formed mapping syntax; must not include any Perforce stream or spec depots, and all depot paths on the right side must match exactly across all branch definitions. You can add and remove only certain types of Perforce branches from this view after you clone the repo.
stream	String	Defines a Perforce stream that maps to the Git branch.
		Provide a stream name using the syntax //streamdepot/ mystream. A Git Fusion branch can be defined as a view or a stream but not both. If your branch is defined as stream, it can include only one stream.
readOnly	String	Prohibit git pushes that introduce commits to the branch.

GitFusionRepoGlobalOverrides

A list of per-repo settings that override global settings.

Table 796. Attributes

Name	Type	Description
charset	String	Defines the default Unicode setting that Git Fusion applies to new repos. This setting is valid only when Git Fusion interacts with a Unicode-enabled Perforce server. (Defaults to UTF-8).
depotPathRepoCreationEnable	String	Allow Git users to create new repos by pushing/pulling a git url which specifies a Perforce

Name	Type	Description
		depot path. This is similar to creating a repo from a p4 client.
depotPathRepoCreationP4group	String	Restrict which authenticated Git pushers are allowed to create new repos when depot-path-repo-creation-enable is enabled.
changeOwner	String	Defines whether Git Fusion assigns either the Git commit author or the Git pusher as the owner of a pushed change (submit).
enableGitBranchCreation	String	Defines whether Git Fusion creates a new branch of Perforce depot file hierarchy for each copied branch of Git workspace history, including Git task branches as Git Fusion anonymous branches.
enableSwarmReviews	String	Permits branch creation for Swarm reviews, even when enable-git-branch-creation is disabled.
enableGitMergeCommits	String	Defines whether Git Fusion copies merge commits and displays them in Perforce as integrations between Perforce branches.
enableGitSubmodules	String	Defines whether Git Fusion allows Git submodules to be pushed to Perforce.
ignoreAuthorPermissions	String	Defines whether Git Fusion evaluates both the author's and pusher's Perforce write permissions during a push or evaluates only the pusher's permissions.
preflightCommit	String	Enables you to trigger pre-flight commit scripts that enforce local policy for Git pushes. This can be especially useful if you have Perforce

Name	Type	Description
		submit triggers that could reject a push and damage the repository.
readPermissionCheck	String	Enables you to require that Git clone, pull, or fetch requests check the Perforce protections table for the puller's read permission on the files being pulled.
gitMergeAvoidanceAfterChange	String	If the Perforce service includes any changelists submitted by Git Fusion 13.2 or earlier, you can prevent unnecessary merge commits by setting this key to the number of the last changelist submitted before your site upgraded to a later version of Git Fusion.
jobLookup	String	Set the format for entering Perforce jobs in Git commit descriptions so that they are recognized by Git Fusion and appear in Perforce changelists as fixes. By default, job IDs whose string starts with "job" (as in job123456) are passed through to the changelist description and job field. Use this option if you want Git Fusion to recognize additional expressions, such as JIRA issue IDs.
depotBranchCreationEnable	String	Allow Git users to create new fully-populated depot branches within Perforce.
depotBranchCreationP4group	String	Restrict the authenticated Git pushers who are allowed to create new fully-populated depot branches, if depotBranchCreationEnable is enabled.
depotBranchCreationDepotPath	String	Tell Git Fusion where to create new fully-

Name	Type	Description
		populated depot branches, if depotBranchCreationEnable is enabled.
		Default path is **//depot/[repo]/[git_branch_name]**.
depotBranchCreationView	String	Set how the depot path set in depotBranchCreationDepotPath should appear in Git.
		Enter a Perforce view specification that maps Perforce depot paths (left side) to Git work tree paths (right side). Perforce depot paths are relative to the root set in depotBranchCreationDepotPath.
		The default ... ... maps every file under the depotBranchCreationDepotPath root to Git. Right side paths must match the right side for every other branch already defined within a repo.
enableGitFindCopies	String	When Git reports a copy file action, store that action in Perforce as a p4 integ. Often set in tandem with enableGitFindRenames.
		No/Off/0%: Do not use Git's copy detection. Treat all possible file copy actions as p4 add actions.
		1%-100%: Use Git's copy detection. Value passed to git diff-tree --find-copies=n.
		Git Fusion also adds --find-copies-harder whenever adding --find-copies.
enableGitFindRenames	String	When Git reports a rename (also called move) file action, store that in Perforce as a p4

Name	Type	Description
		move. Often set in tandem with enableGitFindCopies. `No`/`Off`/`0%`: Do not use Git's rename detection. Treat all possible file rename actions as independent p4 delete and p4 add actions. `1%`-`100%`: Use Git's rename detection. Value passed to git diff-tree --find-renames=n.
enableStreamImports	String	Enables you to convert Perforce stream import paths to Git submodules when you clone a Git Fusion repository. If set to Yes, you must also set either httpUrl or sshUrl.
httpUrl	String	The URL used by Git to clone a repository from Git Fusion over HTTP. This property is required if you want to use Perforce stream import paths as git submodules and you use HTTP(S).
sshUrl	String	The "URL" used by Git to clone a repository from Git Fusion using SSH. This property is required if you want to use Perforce stream import paths as git submodules and you use SSH.
emailCaseSensitivity	String	Defines whether Git Fusion pays attention to case when matching Git user email addresses to Perforce user account email addresses during the authorization check.
authorSource	String	Defines the source that Git Fusion uses to identify the Perforce user associated with a Git push. Defaults to `git-email`.

Name	Type	Description
		Use any one of the following values:
		• **git-email**: Use the email address of the Git author to look for a Perforce user account with the same email address. Git Fusion consults the p4gf_usermap file first, and if that fails to produce a match, it scans the Perforce user table.
		• **git-user**: Use the user.name field in the Git commit. This is the part of the author field before the email address.
		• **git-email-account**: Use the account portion of the Git author's email address. If the Git author's email value is `samwise@the_shire.com`, Git Fusion uses the Perforce account samwise.
		You can also tell Git Fusion to iterate through multiple source types until it finds a matching Perforce account. Specify the source types in order of precedence, separated by commas. For example: git-user, git-email-account, git-email.
limitSpaceMb	String	Natural number representing the number of megabytes of disk space that can be consumed by any single repo. This value does not include the spaced consumed on the Perforce server.
limitCommitsReceived	String	Natural number representing the maximum number of commits allowed in a single push.

Name	Type	Description
limitFilesReceived	String	Natural number representing the maximum number of files allowed in a single push.
limitMegabytesReceived	String	Natural number representing the maximum number of megabytes allowed in a single push.

GroupCommand

Add or delete users from a group, or set the maxresults, maxscanrows, maxlocktime, and timeout limits for the members of a group.

Table 797. Attributes

Name	Type	Description
group	String	The name of the group, as entered on the command line.
maxResults	String	The maximum number of results that members of this group can access from the service from a single command. The default value is **unset**.
maxScanRows	String	The maximum number of rows that members of this group can scan from the service from a single command. The default value is **unset**.
maxLockTime	String	The maximum length of time (in milliseconds) that any one operation can lock any database table when scanning data. The default value is **unset**.
maxOpenFiles	String	The maximum number of files that a member of a group can open using a single command.
timeout	String	The duration (in seconds) of the validity of a session ticket created by p4 login. The default value is 43,200 seconds (12 hours). To create a ticket

Name	Type	Description
		that does not expire, set the Timeout: field to `unlimited`.
passwordTimeout	String	The length of time (in seconds) for which passwords for users in this group remain valid. To disable password aging, use a value of unset.
ldapConfig	String	The LDAP configuration to use when populating the group's user list from an LDAP query.
ldapSearchQuery	String	The LDAP query used to identify the members of the group.
ldapUserAttribute	String	The LDAP attribute that represents the user's username.
subgroups	Array of String	Names of other Perforce groups. To add all users in a previously defined group to the group you're presently working with, include the group name in the Subgroups: field of the p4 group form. Note that user and group names occupy separate namespaces, and thus, groups and users can have the same names. Every member of any previously defined group you list in the Subgroups: field will be a member of the group you're now defining.

owners	Array of String	Names of other Perforce users.
		
		`Group owners without super access are permitted to administer this group, provided that they use the -a option.`
		
		`Group owners are not necessarily members of a group; if a group owner is to`

Name	Type	Description
		be a member of the group, the userid must also be added to the Users: field. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
users	Array of String	The Perforce usernames of the group members.

GroupsCommand

A list of entries that can show the layout how users are associated with the different groups in the system.

Table 798. Attributes

Name	Type	Description
user	String	
group	String	
isSubGroup	String	
isOwner	String	
isUser	String	
maxResults	String	
maxScanRows	String	
maxLockTime	String	
maxOpenFiles	String	
timeout	String	
passTimeout	String	

HWSStatus

Table 799. Attributes

Name	Type	Description
status	String	When "OK" the server should be considered to be operating normally
version	String	The version of Helix Web Services server.

JobCommand

A defect, enhancement request, or other job specification.

The actual fields in a job can be edited by a superuser in your system. The default set of fields in a system are Job, Status, User, Date, and Description.

Table 800. Attributes

Name	Type	Description
Job	String	The job name.

JobsCommand

A summary of jobs in the system.

The actual fields in a job can be edited by a superuser in your system. The default set of fields in a system are Job, Status, User, Date, and Description. Fields in the output of this command may be missing if the superuser removed User, Status, Date, or Description.

Table 801. Attributes

Name	Type	Description
Job	String	The job name.

LabelsCommand

Table 802. Attributes

Name	Type	Description
label	String	The label name.
update	Date	The date the label specification was last modified.

Name	Type	Description
access	Date	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)
owner	String	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
options	String	Options to control behavior and storage location of labels. • locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync. • autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels.
description	String	An optional description of the label's purpose.

LabelCommand

A label specification.

Labels can be either automatic or static. Automatic labels refer to the revisions provided in the View: and Revision: fields. Static labels refer only to those specific revisions tagged by the label by means of either the p4 labelsync or p4 tag commands.

Table 803. Attributes

Name	Type	Description
label	String	The label name.
owner	String	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.
update	Date	The date the label specification was last modified.
access	Date	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)
description	String	An optional description of the label's purpose.
options	String	Options to control behavior and storage location of labels. • locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync.

Name	Type	Description
		• autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels.
revision	String	An optional revision specification for an automatic label. If you use the # character to specify a revision number, you must use quotes around it in order to ensure that the # is parsed as a revision specifier, and not as a comment field in the form.
view	Array of String	A list of depot files that can be tagged with this label. No files are actually tagged until `p4 labelsync` is invoked. Unlike client views or branch views, which map one set of files to another, label views consist of a simple list of depot files.
serverID	String	If set, restricts usage of the label to the named server. If unset, this label may be used on any server.

Location

A consolidated mechanism for identifying something that generally has a path in the system.

Each location references either a depot, a dir, or a file.

Table 804. Attributes

Name	Type	Description
depotPath	String	An absolute depot path specification.
depot	["DepotsCommand" on page 425](#)	
dir	["DirsCommand" on page 427](#)	
file	["FilesCommand" on page 427](#)	
fstat	["FstatCommand" on page 428](#)	
content	String	If this location indicates a single file, this can be set with the Base64-encoded content of the file.

LoginRequest

Captures the login information we need for logging into either a p4d server or our "authentication source".

Table 805. Attributes

Name	Type	Description
user	String	Usually the Perforce username
password	String	
serverLogins	Array of ["ServerLoginRequest" on page 44](#)	

ServerLoginRequest

Table 806. Attributes

Name	Type	Description
id	String	The server's ID
user	String	
password	String	

LoginResponse

Either of our login methods return a ticket, which is then used as a password in a basic authentication scheme.

When this is returned from the explicit p4d login, this is a host unlocked ticket, acceptable for using with a local client.

Table 807. Attributes

Name	Type	Description
ticket	String	

P4dConfigId

Identification of servers the Helix Web Services instance can connect to.

Table 808. Attributes

Name	Type	Description
id	String	A simple string identifier (alphanumeric characters only, please)
name	String	A display string, not guaranteed to be unique
description	String	A simple textual description, for potential selection by clients.

Protections

Displays the information stored in the `p4 protect` command.

Table 809. Attributes

Name	Type	Description
protections	Array of String	Each item in the protections array is a line in the protections table, and is split into five columns.
		1. Access level or mode. One of the access levels list, read, open, write, admin, super, review; or one of the rights =read, =open, =write, and =branch,
		2. Either **user** or **group**, to indicate what's identified by this entry.

Name	Type	Description
		3. The group name or user name. To grant permission to all users, use a wildcard with just an asterix symbol.
		4. The IP address of the client host.
		5. The depot file path, which can contain wildcards. To exclude this mapping from the permission set, use a dash - as the first character of this value.
		IPv6 addresses and IPv4 addresses are also supported. You can use the * wildcard to refer to all IP addresses, but only when you are not using CIDR notation.
		If you use the * wildcard with an IPv6 address, you must enclose the entire IPv6 address in square brackets. For example, [2001:db8:1:2:*] is equivalent to [2001:db8:1:2::]/64. Best practice is to use CIDR notation, surround IPv6 addresses with brackets, and to avoid the * wildcard.
		How the system forms host addresses depends on the setting of the dm.proxy.protects variable. By default, this variable is set to 1. This means that if the client host uses some intermediary (proxy, broker, replica) to access the server, the proxy- prefix is prepended to the client host address to indicate that the connection is not direct. If you specify proxy-* for the Host field, that will affect all connections made via proxies, brokers, and replicas.

Name	Type	Description
		A value like proxy-10.0.0.5 identifies a client machine with an IP address of 10.0.0.5 that is connected to the server through an intermediary.
		Setting the dm.proxy.protects variable to 0, removes the proxy- prefix and allows you to write a single set of protection entries that apply both to directly-connected clients as well as to those that connect via an intermediary. This is more convenient but less secure if it matters that a connection is made using an intermediary. If you use this setting, all intermediaries must be at release 2012.1 or higher.

ServersCommand

Table 810. Attributes

Name	Type	Description
serverID	String	A unique identifier for this server. This must match the contents of the server's `server.id` file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.
type	String	Server executable type. One of the following: `server`, `proxy`, `broker`, `identifier`, `admin`. Each type may offer one or more services, defined in the `services` property.
services	String	The `server` type server provides the following services:

Name	Type	Description
		• standard - a standard Perforce server
		• replica - a read-only replica server
		• commit-server - central server in distributed installation
		• edge-server - node in distributed installation
		• forwarding-replica - a replica configured to forward commands that involve database writes to a master server
		• build-server - a replica that supports build automation and build farm integration
		• P4AUTH - a server that provides authentication
		• P4CHANGE - a server that provides change numbering
		• depot-master - commit-server with automated failover
		• depot-standby - standby replica of the depot-master
		• workspace-server - node in a cluster installation
		• standby - read-only replica server that uses p4 journalcopy
		• forwarding-standby - forwarding replica server that uses p4 journalcopy
		The **proxy** type server provides a p4p caching proxy.
		The **broker** type server provides the following services:

Name	Type	Description
		• broker - a p4broker process
		• workspace-router - routing broker for a cluster
		The services field for the `identifier` type server specifies the existence of the cluster, and has the value `cluster`. The name of the cluster is then drawn from the ServerID field.
		The `admin` type server provides the following services:
		• hxca-server - the admin server for a Helix cluster.
		• zookeeper-server - ZooKeeper server for a cluster
name	String	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.
address	String	The P4PORT used by this server.
description	String	An optional description for this server.
user	String	The service user name used by the server.

ServerCommand

The Perforce server specification describes the high-level configuration and intended usage of a Perforce server. For installations with only one Perforce server, the server specification is optional.

Table 811. Attributes

Name	Type	Description
serverID	String	A unique identifier for this server. This must match the contents of the server's `server.id` file as defined by

Name	Type	Description
		the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.
type	String	Server executable type. One of the following: `server`, `proxy`, `broker`, `identifier`, `admin`. Each type may offer one or more services, defined in the `services` property.
services	String	The `server` type server provides the following services: • standard - a standard Perforce server • replica - a read-only replica server • commit-server - central server in distributed installation • edge-server - node in distributed installation • forwarding-replica - a replica configured to forward commands that involve database writes to a master server • build-server - a replica that supports build automation and build farm integration • P4AUTH - a server that provides authentication • P4CHANGE - a server that provides change numbering • depot-master - commit-server with automated failover

Name	Type	Description
		• depot-standby - standby replica of the depot-master
		• workspace-server - node in a cluster installation
		• standby - read-only replica server that uses p4 journalcopy
		• forwarding-standby - forwarding replica server that uses p4 journalcopy
		The **proxy** type server provides a p4p caching proxy.
		The **broker** type server provides the following services:
		• broker - a p4broker process
		• workspace-router - routing broker for a cluster
		The services field for the **identifier** type server specifies the existence of the cluster, and has the value **cluster**. The name of the cluster is then drawn from the ServerID field.
		The **admin** type server provides the following services:
		• hxca-server - the admin server for a Helix cluster.
		• zookeeper-server - ZooKeeper server for a cluster
name	String	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.
address	String	The P4PORT used by this server.

Name	Type	Description
externalAddress	String	For an edge server, this optional field specifies the external address used for connections to a commit server. This field must be set for the edge server to enable parallel submits in a federated environment.
description	String	An optional description for this server.
user	String	The service user name used by the server.
clientDataFilter	String	For a replica server, this optional field can contain one or more patterns describing how active client workspace metadata is to be filtered. Active client workspace data includes have lists, working records, and pending resolves. To include client data, use the syntax: **//client-pattern/…** To exclude client data, use the syntax: **-//client-pattern/…** All patterns are specified in client syntax.
revisionDataFilter	String	For a replica server, this optional field can contain one or more patterns describing how submitted revision metadata is to be filtered. Submitted revision data includes revision records, integration records, label contents, and the files listed in submitted changelists. To include depot data, use the syntax: To exclude depot data, use the syntax: -//depot/pattern/…

Name	Type	Description
		All patterns are specified in depot syntax.
archiveDataFilter	String	For a replica server, this optional field can contain one or more patterns describing the policy for automatically scheduling the replication of file content. If this field is present, only those files described by the pattern are automatically transferred to the replica; other files are not transferred until they are referenced by a replica command that needs the file content.
		Files specified in the ArchiveDataFilter: field are transferred to the replica regardless of whether any users of the replica have made requests for their content.
		To automatically transfer files on submit, use the syntax: `//depot/pattern/…`
		To exclude files from automatic transfer, use the syntax: `-//depot/pattern/…`
		All patterns are specified in depot syntax.
distributedConfig	String	For an edge or commit server, this optional field, which is displayed only when you use the -l or -c option, shows configuration settings for this server.
		`-l` flag shows the current configuration. `-c-` flag shows current configuration values, recommended default values for fields that are not set, or

Name	Type	Description
		unset for fields that are not set and do not have default values.
		If this field is present when invoked with -c, the configuration commands in this field are run on the current server using the scope of the server specified in the serverID field.

StreamCommand

The Perforce stream specification defines a single stream.

Streams are hierarchical branches with policies that control the structure and the flow of change. Stream hierarchies are based on the stability of the streams, specified by the type you assign to the stream. Development streams are least stable (most subject to change), mainline streams are somewhat stable, and release streams are highly stable. Virtual streams can be used to copy and merge between parent and child streams without storing local data. Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data.

Stream contents are defined by the paths that you map. By default, a stream has the same structure as its parent (the stream from which it was branched), but you can override the structure, for example to ensure that specified files cannot be submitted or integrated to other streams.

Table 812. Attributes

Name	Type	Description
stream	String	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form **//depotname/streamname**.
update	Date	The date the stream specification was last modified.
access	Date	The date and time that the stream specification was last accessed by any Perforce command.
owner	String	The Perforce user or group who owns the stream. The default is the user who created the stream.
name	String	Display name of the stream. Unlike the Stream: field, this

Name	Type	Description
		field can be modified. Defaults to the streamname portion of the stream path.
parent	String	The parent of this stream. Must be none if the stream's `Type:` is mainline, otherwise must be set to an existing stream identifier of the form `//depotname/streamname`.
type	String	The stream's type determines the expected flow of change. Valid stream types are `mainline`, `virtual`, `development`, and `release`. • `mainline`: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream. • `virtual`: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream. • `release`: More stable than the mainline. Release streams copy from the parent and merge to the parent. • `development`: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent.

Name	Type	Description
		• **task**: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.
description	String	Description of the stream.
options	String	Settings that configure stream behavior as follows: • **[un]locked**: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification cannot be deleted, and only its owner can modify it. The default is unlocked. • **[all,owner]submit**: Specifies whether all users or only the owner of the stream can submit changes to the stream. The default is allsubmit. If the Owner: of a stream marked ownersubmit is a group, all users who are members of that group can submit changes to the stream. • **[no]toparent**: Specifies whether integrations from the stream to its parent are

Name	Type	Description
		expected. The default is toparent.
		• [no]fromparent: Specifies whether integrations to the stream from its parent are expected. The default is fromparent for mainline and development streams, and nofromparent for release streams.
		• mergeany,mergedown: Specifies whether the merge flow is restricted or whether merge is permitted from any other stream. For example, the mergeany option would allow a merge from a child to a parent with no warnings. A virtual stream must have its flow options set to notoparent and nofromparent. Flow options are ignored for mainline streams.
paths	Array of String	Paths define how files are incorporated into the stream structure. Specify paths using the following format: path_type view_path [depot_path] where path_type is a single keyword, view_path is a file path with no leading slashes, and the optional depot_path is a file path beginning with //.
		The default path is share …
		Valid path types are:
		• share view_path: Specified files can be synced, submitted, and integrated to and from the parent stream.
		• isolate view_path: Specified files can be synced and

Name	Type	Description
		submitted, but cannot be integrated to and from the parent stream.
		• **import** `view_path` **[depot_path]**: Specified files can be synced, but cannot be submitted or integrated to and from the parent stream. The **view_path** is mapped as in the parent stream's view, or to an (optional) **depot_path**. The **depot_path** may include a changelist specifier. That stream's client workspaces will be limited to seeing revisions at that change or lower within that depot path. For example, you can specify a depot path like this: **//depot/import/…@1000**. Revisions from changelists greater than 1000 will be automatically hidden from most commands. The changelist limits in effect for a given stream workspace are displayed in a read-only client workspace specification field called ChangeView.
		• **import+** `view_path` **[depot_path]**: Functions like a standard import path, enabling you to map a path from outside the stream depot to your stream, but unlike a standard import path, you can submit changes to the files in an import+ path.
		• **exclude** `view_path`: Specified files cannot be synced, submitted or integrated to and from the parent stream. By default, streams inherit their structure from

Name	Type	Description
		the parent stream (except mainlines, which have no parent). Paths are inherited by child stream views; a child stream's path can downgrade the inherited view, but not upgrade it. (For example, a child stream can downgrade a shared path to an isolated path, but if the parent stream defines a path as isolated, its child cannot restore full access by specifying the path as shared.) Note that the `depot_path` is relevant only when the path_type is `import` or `import+`.
remapped	Array of String	Reassigns the location of workspace files. To specify the source path and its location in the workspace, use the following syntax: `view_path_1 view_path_2` where `view_path_1` and `view_path_2` are Perforce view paths (omit leading slashes and leading or embedded wildcards; terminal wildcards are fine). For example, to ensure that files are synced to the local ProjectX folder, remap as follows: … `projectX/`… Line ordering in the Remapped: field is significant: if more than one line remaps the same files, the later line takes precedence. Remappings are inherited by child streams and the workspaces associated with them.
ignored	Array of String	A list of file or directory names to be ignored in client views. For example:

Name	Type	Description

		`/tmp # ignores files named "tmp"`
		`/tmp/... # ignores directories named "tmp"`
		`.tmp # ignores file names ending in .tmp`
		
		Lines in the `Ignored:` field can appear in any order. Ignored files and directories are inherited by child stream client views.

StreamsCommand

A summary of a stream in the system, as provided by the `p4 streams` command.

Table 813. Attributes

Name	Type	Description
stream	String	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form **//depotname/streamname**.
update	Date	The date the stream specification was last modified.
access	Date	The date and time that the stream specification was last accessed by any Perforce command.
owner	String	The Perforce user or group who owns the stream. The default is the user who created the stream.
name	String	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname portion of the stream path.
parent	String	The parent of this stream. Must be none if the stream's **Type:** is mainline, otherwise must be set to an existing stream identifier

Name	Type	Description
		of the form **//depotname/ streamname**.
type	String	The stream's type determines the expected flow of change. Valid stream types are `mainline`, `virtual`, `development`, and `release`.
		• `mainline`: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream.
		• `virtual`: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream.
		• `release`: More stable than the mainline. Release streams copy from the parent and merge to the parent.
		• `development`: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent.
		• `task`: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched

Name	Type	Description
		(copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.
description	String	Description of the stream.
options	String	Settings that configure stream behavior as follows: • `[un]locked`: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification cannot be deleted, and only its owner can modify it. The default is unlocked. • `` `[all ``

Triggers

Defines the triggers table, like it would appear in the output to the `p4 triggers` command.

Table 814. Attributes

Name	Type	Description
triggers	Array of String	A list of trigger definitions. A trigger definition contains four fields that specify the name of the trigger, the type of event that should trigger the execution of the script, the paths that should be affected by the trigger, the location of the script, and other trigger type-dependent information.

Name	Type	Description
		When the condition specified in a trigger definition is satisfied, the associated script or program is executed. Example: `trig1 change-submit //depot/dir/… "/usr/bin/s1.pl %changelist%"` See the Helix Versioning Engine Administrator Guide for more details on trigger definitions.

UserCommand

Create or edit Perforce user specifications and preferences.

There are three types of Perforce users: standard users, operator users, and service users. Standard users are the default, and each standard user consumes one Perforce license. The operator user type is intended for system administrators; they are subject to the same restrictions on permissions as any other user, but are further restricted in that they can run only a limited subset of Perforce commands. Service users are intended for inter-server communication in replicated and multi-server environments, and are restricted to an even smaller subset of Perforce commands. Neither operators nor service users consume Perforce licenses.

Table 815. Attributes

Name	Type	Description
user	String	The Perforce username.
type	String	Type of user: standard, operator, or service. Once you set the type, you cannot change it.
authMethod	String	One of the following: perforce or ldap. Specifying perforce enables authentication using Perforce's internal db.user table or by way of an authentication trigger. This is the default unless it is overridden with the auth.default.method configurable.

Name	Type	Description
		Specifying ldap enables authentication against AD/LDAP servers specified by the currently active LDAP configurations.
email	String	The user's email address. By default, this is user@client.
update	Date	The date and time this specification was last updated.
access	Date	The date and time this user last ran a Perforce command.
fullName	String	The user's full name.
jobView	String	Jobs matching this jobview appear on any changelists created by this user. Jobs that are fixed by the changelist should be left in the changelist when it's submitted with p4 submit; other jobs should be deleted from the form before submission.
password	String	The user's password.
passwordChange	Date	The date and time of the user's last password change. If the user has no password, this field is blank.
reviews	Array of String	A list of files the user would like to review. This field can include exclusionary mappings.

UsersCommand

Table 816. Attributes

Name	Type	Description
user	String	The Perforce username.
type	String	Type of user: standard, operator, or service.

Name	Type	Description
		Once you set the type, you cannot change it.
email	String	The user's email address. By default, this is user@client.
update	Date	The date and time this specification was last updated.
access	Date	The date and time this user last ran a Perforce command.
fullName	String	The user's full name.
hasPassword	String	If 'enabled', the password has been set on the user.

HTTP Method Reference

Default Methods

Most paths start with the platform version in the URL.

GET /api/2016.1.0/config/p4ds

Description

The list of registered p4d servers in your cluster.

This is provided by a special set of configuration files in the system. For more information, consult the Helix Web Services user guide.

Responses

HTTP Code	Description	Schema
200		["P4dConfigId" on page 105]

POST /api/2016.1.0/login

Description

Logs into the primary authentication source.

This can either be a p4d instance or Helix Cloud, depending upon the configuration of your Helix Web Services instance.

Parameters

Where	Name	Description	Required	Schema	Default
body	loginRequest	The user login and password.	true	"LoginRequest"	

Responses

HTTP Code	Description	Schema
200	Object with ticket to use for Basic auth password.	"LoginResponse" on page 105

GET /api/2016.1.0/status

Description

A simple structure to monitor for "problems" an admin should take care of, and, report the current application version.

This method does not require authentication.

Responses

HTTP Code	Description	Schema
200		"HWSStatus" on page 101

GET /api/2016.1.0/{server}/branches

Description

Lists available branches in the system. The resources of this list are summaries of branches in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	Summaries of branches in the system.	["BranchesCommand" on page 76]

POST /api/2016.1.0/{server}/branches

Description

Creates a new branch specification, like the `p4 branch` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	body	The branch specification.	true	"BranchComma	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/branches/{branch}

Description

Returns the branch spec details of the particular branch.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular	true	string	

Where	Name	Description	Required	Schema	Default
		method against.			
path	branch	The branch ID	true	string	

Responses

HTTP Code	Description	Schema
200	Branch spec details	"BranchCommand" on page 75

PATCH /api/2016.1.0/{server}/branches/{branch}

Description

Update branch specifications, similar to the `p4 branch` command. Only the specified parameters in the body will be changed.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	branch	The branch ID	true	string	
body	body	Fields of the branch to update	true	"BranchComma	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

DELETE /api/2016.1.0/{server}/branches/{branch}

Description

Removes the branch specification, similar to the `p4 branch -d` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	branch	The branch ID	true	string	

Responses

HTTP Code	Description	Schema
200		["CommandResponse" on page 87](#)

GET /api/2016.1.0/{server}/changes

Description

Lists available changes in the system. The resources of this list are summaries of changes in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
query	max	Limit the number of change results	false	integer	
query	status	The status of the changes, e.g., `submitted`	false	string	
query	user	The user's login who submitted the change	false	string	

Where	Name	Description	Required	Schema	Default
query	files	Limit changes to the depot path expressions. See the [changes](#) command description.	false	string	

Responses

HTTP Code	Description	Schema
200	Summaries of changes in the system.	[["ChangesCommand" on page 79](#)]

GET /api/2016.1.0/{server}/changes/{change}

Description

Returns the change spec details of the particular change.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	change	The change ID	true	string	

Responses

HTTP Code	Description	Schema
200	Change spec details	["ChangeCommand" on page 76](#)

GET /api/2016.1.0/{server}/clients

Description

Lists available clients in the system. The resources of this list are summaries of clients in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	Summaries of clients in the system.	["ClientsCommand" on page 84]

POST /api/2016.1.0/{server}/clients

Description

Creates a new client specification, like the `p4 client` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	client	The client spec	true	"ClientCommar	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/clients/{client}

Description

Returns the client spec details of the particular client.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	client	The client ID	true	string	

Responses

HTTP Code	Description	Schema
200	Client spec details	["ClientCommand" on page 80](#)

PATCH /api/2016.1.0/{server}/clients/{client}

Description

Update client specifications, similar to the `p4 client` command. Only the specified parameters in the body will be changed.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	client	The client ID	true	string	
body	body	Fields of the client to update	true	["ClientCommar](#)	

Responses

HTTP Code	Description	Schema
200		["CommandResponse" on page 87](#)

DELETE /api/2016.1.0/{server}/clients/{client}

Description

Removes the client specification, similar to the `p4 client -d` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	client	The client ID	true	string	

Responses

HTTP Code	Description	Schema
200		[“CommandResponse” on page 87](#)

GET /api/2016.1.0/{server}/commands/{command}

Description

Execute a Perforce command that requires no input. This only allows commands that have been whitelisted on your system. See the [“Configuration” on page 23](#) section for details.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	command	The command name	true	string	
query	arg	Command arguments	false	array	

Responses

HTTP Code	Description	Schema
200	Generic list of hashes response	["CommandResponse" on page 87](#)

POST /api/2016.1.0/{server}/commands/{command}

Description

Execute a Perforce command that accepts input, like a spec. This only allows commands that have been whitelisted on your system. See the ["Configuration" on page 23](#) section for details.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	command	The command name	true	string	
query	arg	Command arguments	false	array	
body	input	A hash used as input to the command	false	["CommandReq](#)

Responses

HTTP Code	Description	Schema
200	Generic list of hashes response	["CommandResponse" on page 87](#)

GET /api/2016.1.0/{server}/counters

Description

Lists available counters in the system. The resources of this list are summaries of counters in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	Summaries of counters in the system.	["Counter" on page 88]

PUT /api/2016.1.0/{server}/counters/{counter}

Description

Update counter specifications, similar to the `p4 counter` command. Only the specified parameters in the body will be changed.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	counter	The counter ID	true	string	
body	body	Fields of the counter to update	true	"Counter" on pa	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/counters/{counter}

Description

Returns the counter spec details of the particular counter.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	counter	The counter ID	true	string	

Responses

HTTP Code	Description	Schema
200	Counter spec details	"Counter" on page 88

DELETE /api/2016.1.0/{server}/counters/{counter}

Description

Removes the counter specification, similar to the `p4 counter -d` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	counter	The counter ID	true	string	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

POST /api/2016.1.0/{server}/counters/{counter}/increment

Description

Increments a numerical counter, similar to the `p4 counter -i` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	counter	The counter ID	true	string	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/depots

Description

Lists available depots in the system. The resources of this list are summaries of depots in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we	true	string	

Where	Name	Description	Required	Schema	Default
		execute this particular method against.			

Responses

HTTP Code	Description	Schema
200	Summaries of depots in the system.	["DepotsCommand" on page 90]

POST /api/2016.1.0/{server}/depots

Description

Creates a new depot specification, like the `p4 depot` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	depot	The depot spec	true	"DepotCommar	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/depots/{depot}

Description

Returns the depot spec details of the particular depot.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	depot	The depot ID	true	string	

Responses

HTTP Code	Description	Schema
200	Depot spec details	["DepotCommand" on page 88](#)

PATCH /api/2016.1.0/{server}/depots/{depot}

Description

Update depot specifications, similar to the `p4 depot` command. Only the specified parameters in the body will be changed.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	depot	The depot ID	true	string	
body	body	Fields of the depot to update	true	["DepotCommar](#)	

Responses

HTTP Code	Description	Schema
200		["CommandResponse" on page 87](#)

DELETE /api/2016.1.0/{server}/depots/{depot}

Description

Removes the depot specification, similar to the `p4 depot -d` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	depot	The depot ID	true	string	

Responses

HTTP Code	Description	Schema
200		["CommandResponse" on page 87](#)

GET /api/2016.1.0/{server}/groups

Description

Lists available groups in the system. The resources of this list are summaries of groups in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	Summaries of groups in the system.	[["GroupsCommand" on page 100](#)]

POST /api/2016.1.0/{server}/groups

Description

Creates a new group specification, like the **p4 group** command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	body	The group spec	true	"GroupCommaı	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/groups/{group}

Description

Returns the group spec details of the particular group.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	group	The group ID	true	string	

Responses

HTTP Code	Description	Schema
200	Group spec details	"GroupCommand" on page 99

PATCH /api/2016.1.0/{server}/groups/{group}

Description

Update group specifications, similar to the `p4 group` command. Only the specified parameters in the body will be changed.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	group	The group ID	true	string	
body	body	Fields of the group to update	true	"GroupComman	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

DELETE /api/2016.1.0/{server}/groups/{group}

Description

Removes the group specification, similar to the `p4 group -d` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	group	The group ID	true	string	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/jobs

Description

Lists available jobs in the system. The resources of this list are summaries of jobs in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	Summaries of jobs in the system.	["JobsCommand" on page 101]

POST /api/2016.1.0/{server}/jobs

Description

Creates a new job specification, like the `p4 job` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Where	Name	Description	Required	Schema	Default
body	job	The job spec	true	"JobCommand"	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/jobs/{job}

Description

Returns the job spec details of the particular job.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	job	The job ID	true	string	

Responses

HTTP Code	Description	Schema
200	Job spec details	"JobCommand" on page 101

PATCH /api/2016.1.0/{server}/jobs/{job}

Description

Update job specifications, similar to the `p4 job` command. Only the specified parameters in the body will be changed.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we	true	string	

Where	Name	Description	Required	Schema	Default
		execute this particular method against.			
path	job	The job ID	true	string	
body	jobCommand	Fields of the job to update	true	"JobCommand"	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

DELETE /api/2016.1.0/{server}/jobs/{job}

Description

Removes the job specification, similar to the `p4 job -d` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	job	The job ID	true	string	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

DELETE /api/2016.1.0/{server}/jobs/{job}/fixes/{change}

Description

Removes the fix record association for the job for a particular changelist.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	job	The job ID	true	string	
path	change	The change ID	true	string	

Responses

HTTP Code	Description	Schema
200		[“CommandResponse” on page 87](#)

POST /api/2016.1.0/{server}/jobs/{job}/fixes/{change}

Description

Adds a fix record to the job for a particular changelist.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	job	The job ID	true	string	
path	change	The change ID	true	string	
query	status	Specify the job status instead of using the default. The default is	false	string	

Where	Name	Description	Required	Schema	Default
		typically **closed** or some other value defined in the **Presets** field specified in the **p4 jobspec** form. If the changelist to which you're linking the job been submitted, the status value is immediately reflected in the job's status. If the changelist is pending, the job status is changed on submission of the changelist, provided that the -s option is also supplied to p4 submit and the desired status appears next to the job in the p4 submit form's Jobs: field. To leave a job unchanged, use the special status of same.			

Responses

HTTP Code	Description	Schema
200		["CommandResponse" on page 87](#)

GET /api/2016.1.0/{server}/labels

Description

Lists available labels in the system. The resources of this list are summaries of labels in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	Summaries of labels in the system.	["LabelsCommand" on page 102]

POST /api/2016.1.0/{server}/labels

Description

Creates a new label specification, like the `p4 label` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	label	The label spec	true	"LabelCommand	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/labels/{label}

Description

Returns the label spec details of the particular label.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	label	The label ID	true	string	

Responses

HTTP Code	Description	Schema
200	Label spec details	"LabelCommand" on page 103

PATCH /api/2016.1.0/{server}/labels/{label}

Description

Update label specifications, similar to the `p4 label` command. Only the specified parameters in the body will be changed.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	label	The label ID	true	string	
body	labelCommand	Fields of the label to update	true	"LabelComman	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

DELETE /api/2016.1.0/{server}/labels/{label}

Description

Removes the label specification, similar to the `p4 label -d` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	label	The label ID	true	string	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

POST /api/2016.1.0/{server}/login

Description

Logs into a Helix Versioning Engine (p4d) server.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	body	The user login and password.	true	"LoginRequest"	

Responses

HTTP Code	Description	Schema
200	Object with ticket to use for Basic auth password.	["LoginResponse" on page 105](#)

GET /api/2016.1.0/{server}/paths

Description

Lists depots, files, and directories in the system. This combines the output of the `p4 depots`, `p4 dirs`, and `p4 files` commands, depending upon your path.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
query	path	The path "under a depot" to query under, e.g., `//depot/main`. This will list the directories and files underneath that path.	false	string	

Responses

HTTP Code	Description	Schema
200	Array of depots.	[["Location" on page 104](#)]

PUT /api/2016.1.0/{server}/protections

Description

Updates the protections table.

This method requires superuser access.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	protections	The new protections table	true	"Protections" or	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/protections

Description

Returns a list of available protections in the system. The elements of this list are rows of the system's protections table.

This method requires superuser access.

See the output of **p4 protect** for more information.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	Object including list of protections entries	"Protections" on page 106

GET /api/2016.1.0/{server}/servers

Description

Lists available servers in the system. The resources of this list are summaries of servers in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	Summaries of servers in the system.	["ServersCommand" on page 107]

POST /api/2016.1.0/{server}/servers

Description

Creates a new server specification, like the `p4 server` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	serverCommand	The server spec	true	"ServerCommar	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/servers/{serverId}

Description

Returns the server spec details of the particular server.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	serverId	The server ID of the server spec	true	string	

Responses

HTTP Code	Description	Schema
200	Server spec details	"ServerCommand" on page 109

PATCH /api/2016.1.0/{server}/servers/{serverId}

Description

Update server specifications, similar to the `p4 server` command. Only the specified parameters in the body will be changed.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	serverId	The server ID	true	string	
body	serverCommand	Fields of the server to update	true	"ServerComman	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

DELETE /api/2016.1.0/{server}/servers/{serverId}

Description

Removes the server specification, similar to the `p4 server -d` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	serverId	The server ID	true	string	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/streams

Description

Lists available streams in the system. The resources of this list are summaries of streams in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	Summaries of streams in the system.	["StreamsCommand" on page 116]

POST /api/2016.1.0/{server}/streams

Description

Creates a new stream specification, like the `p4 stream` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	body	The stream spec	true	"StreamComma	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/streams/stream

Description

Returns the stream spec details of the particular stream.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular	true	string	

Where	Name	Description	Required	Schema	Default
		method against.			
query	stream	The stream ID	true	string	

Responses

HTTP Code	Description	Schema
200	Stream spec details	"StreamCommand" on page 112

PATCH /api/2016.1.0/{server}/streams/stream

Description

Update stream specifications, similar to the `p4 stream` command. Only the specified parameters in the body will be changed.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
query	stream	The stream ID	true	string	
body	body	Fields of the stream to update	true	"StreamComma	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

DELETE /api/2016.1.0/{server}/streams/stream

Description

Removes the stream specification, similar to the `p4 stream -d` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
query	stream	The stream ID	true	string	

Responses

HTTP Code	Description	Schema
200		[“CommandResponse” on page 87](#)

PUT /api/2016.1.0/{server}/triggers

Description

Updates the triggers table.

This method requires superuser access.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	triggers	The new triggers table	true	[“Triggers” on pa](#)	

Responses

HTTP Code	Description	Schema
200		[“CommandResponse” on page 87](#)

GET /api/2016.1.0/{server}/triggers

Description

Returns a list of available triggers in the system. The elements of this list are rows of the system's triggers table.

This method requires superuser access.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	List of triggers entries	"Triggers" on page 118

GET /api/2016.1.0/{server}/users

Description

Lists available users in the system. The resources of this list are summaries of users in the system.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
query	includeService	If true, shows service users in the list.	false	boolean	
query	max	Cap the number of users	false	integer	

Where	Name	Description	Required	Schema	Default
		reported to this amount.			

Responses

HTTP Code	Description	Schema
200	Summaries of users in the system.	["UsersCommand" on page 120]

POST /api/2016.1.0/{server}/users

Description

Creates a new user specification, like the `p4 user` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	body	The user spec	true	"UserCommand	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/users/{user}

Description

Returns the user spec details of the particular user.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we	true	string	

Where	Name	Description	Required	Schema	Default
		execute this particular method against.			
path	user	The user ID	true	string	

Responses

HTTP Code	Description	Schema
200	User spec details	[“UserCommand” on page 119](#)

PATCH /api/2016.1.0/{server}/users/{user}

Description

Update user specifications, similar to the `p4 user` command. Only the specified parameters in the body will be changed.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	user	The user ID	true	string	
body	body	Fields of the user to update	true	[“UserCommand](#)	

Responses

HTTP Code	Description	Schema
200		[“CommandResponse” on page 87](#)

DELETE /api/2016.1.0/{server}/users/{user}

Description

Removes the user specification, similar to the `p4 user -d` command.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	user	The user ID	true	string	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

Alpha Methods

The following methods are in an alpha state, which means that they are not considered stable and may change in future versions. If you choose to work with these methods your client code is not guaranteed to work in the future.

POST /api/2016.1.0/{server}/changes

Description

Create a new changelist that can affect multiple files using different kinds of actions. If you require the ability to integrate or move, for example, you can use this method.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
body	changelistReque	Description of changes to make	true	"ChangelistReqi

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/git-fusion-repos

Description

Lists all configured repositories readable by the current user. ===== Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	

Responses

HTTP Code	Description	Schema
200	List of configured repository names and IDs	["GitFusionRepoId" on page 94]

POST /api/2016.1.0/{server}/git-fusion-repos

Description

Submits a [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file to create or update a repository configuration.

If the repository does not exist or has been previously deleted, this method saves contents of the config file to a new `p4gf_config` file. If the repository has already been initialised, this method replaces all of the file contents of the specified repository's `p4gf_config` file.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular	true	string	

Where	Name	Description	Required	Schema	Default
		method against.			
body	body	The new configuration	true	"GitFusionRepo	94

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

GET /api/2016.1.0/{server}/git-fusion-repos/{repo}

Description

Return configuration for the specified repository. Grabs and returns contents of the [p4gf_config](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j) file for given repository.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	repo	The Git Fusion Repo ID	true	string	

Responses

HTTP Code	Description	Schema
200	Git Fusion repository config	"GitFusionRepoConfig" on page 94

PATCH /api/2016.1.0/{server}/git-fusion-repos/{repo}

Description

Updates values in the repository configuration. This method will find all specified parameters and update each value for the specified repository's configuration file.

Parameters

Where	Name	Description	Required	Schema	Default	
path	server	The server ID that we execute this particular method against.	true	string		
path	repo	The Git Fusion Repo ID	true	string		
body	body	The new configuration	true	"GitFusionRepo		94

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

DELETE /api/2016.1.0/{server}/git-fusion-repos/{repo}

Description

Deletes the repository configuration (The [p4gf_config file](http://www.perforce.com/perforce/r15.1/manuals/git-fusion/chapter_dyn_ngj_3l.html#section_jgz_nz2_2j)). Contents of the repository are not deleted from Perforce.

Parameters

Where	Name	Description	Required	Schema	Default
path	server	The server ID that we execute this particular method against.	true	string	
path	repo	The Git Fusion Repo ID	true	string	

Responses

HTTP Code	Description	Schema
200		"CommandResponse" on page 87

JSON Definitions

BranchCommand

Name	Description	Required	Schema	Default
branch	The branch name, as provided on the command line.	false	string	
owner	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.	false	string	
access	The date the branch mapping was last accessed.	false	string	
update	The date the branch mapping was last changed.	false	string	
options	Either unlocked (the default) or locked. If locked, only the Owner: can modify the branch mapping, and the mapping can't be deleted until it is unlocked.	false	string	
description	A short description of the branch's purpose.	false	string	
view	A set of mappings from one set of files in the depot	false	[string]	

Name	Description	Required	Schema	Default
	(the source files) to another set of files in the depot (the target files). The view maps from one location in the depot to another; it can't refer to a client workspace. For example, the branch view `//depot/main/…` `//depot/r2.1/…` maps all the files under `//depot/main` to `//depot/r2.1`.			

BranchesCommand

Name	Description	Required	Schema	Default
branch	The branch name, as provided on the command line.	false	string	
owner	The owner of the branch mapping. By default, this will be set to the user who created the branch. This field is unimportant unless the Option: field value is locked.	false	string	
access	The date the branch mapping was last accessed.	false	string	
update	The date the branch mapping was last changed.	false	string	

Name	Description	Required	Schema	Default
options	Either `unlocked` (the default) or `locked`. If `locked`, only the `Owner:` can modify the branch mapping, and the mapping can't be deleted until it is `unlocked`.	false	string	
description	A short description of the branch's purpose.	false	string	

ChangeCommand

Name	Description	Required	Schema	Default
change	Contains the changelist number if editing an existing changelist, or `new` if creating a new changelist.	false	string	
client	Name of current client workspace	false	string	
date	Date the changelist was last modified.	false	string	
user	Name of the change owner. The owner of an empty pending changelist (that is, a pending changelist without any files in it) can transfer ownership of the changelist to	false	string	

Name	Description	Required	Schema	Default
	another existing user either by editing this field, or by using the -U user option. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.			
status	pending, shelved, submitted, or new. Not editable by the user. The status is new when the changelist is created, pending when it has been created but has not yet been submitted to the depot, shelved when its contents are shelved, and submitted when its contents have been stored in the depot.	false	string	
description	Textual description of changelist. If you do not have access	false	string	

Name	Description	Required	Schema	Default
	to a restricted changelist, the description is replaced with a "no permission" message.			
jobs	A list of jobs that are fixed by this changelist.	false	[string]	
type	Type of change: `restricted` or `public`. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change. Public changes are displayed without restrictions. By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the	false	string	

Name	Description	Required	Schema	Default
	defaultChangeType configurable.			
files	The list of files submitted in this changelist.	false	[string]	
importedBy	Displays the name of the user who ran the p4 fetch, p4 push, or p4 unzip command that imported this change into the server. This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied. In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.	false	string	
identify	Contains a label which uniquely identifies this changelist across	false	string	

Name	Description	Required	Schema	Default
	all servers where it has been fetched, pushed, or unzipped. This field is primarily useful for distributed versioning (DVCS) scenarios, in which changelists are copied from one server to another, and help you correlate the changelist's basic identity as it is copied. In such configurations, Perforce recommends using the submit.identity configurable to enable automatic generation of changelist identities by the p4 submit.			

ChangesCommand

Name	Description	Required	Schema	Default
change	The changelist ID	false	string	
date	Last modification time of the changelist	false	string	
user	The owner of the changelist	false	string	
client	Name of current client workspace.	false	string	

Name	Description	Required	Schema	Default
status	pending, shelved, submitted, or new. Not editable by the user. The status is new when the changelist is created, pending when it has been created but has not yet been submitted to the depot, shelved when its contents are shelved, and submitted when its contents have been stored in the depot.	false	string	
type	Type of change: restricted or public. The Type: field can be used to hide the change or its description from users. A shelved or committed change (as denoted in the Status: field) that is restricted is accessible only to users who own the change or have list permission to at least one file in the change. Public changes are displayed without restrictions.	false	string	

Name	Description	Required	Schema	Default
	By default, changelists are public. A Perforce superuser can set the default changelist type (for changelists created after the configurable is set) by setting the defaultChangeType configurable.			
path	Depot paths affected by this changelist	false	string	
description	Textual description of changelist. If you do not have access to a restricted changelist, the description is replaced with a "no permission" message.	false	string	

ChangelistRequest

Name	Description	Required	Schema	Default
description		false	string	
stream	Optional stream ID to use in case you want to edit files in a stream.	false	string	
actions		false	[*"ChangelistAction"*	

ChangelistAction

Name	Description	Required	Schema	Default
depotFile	The target file path to edit.	false	string	
fromDepotFile	For "branch" or "move" actions, this indicates the source file location.	false	string	
actionType	One of "upload", "branch", "move", or "delete"	false	string	
content	Base64-encoded content	false	string	
requireVersion	If set, we will only operate if this is the current version of the file.	false	integer	

ClientCommand

Name	Description	Required	Schema	Default
client	The client workspace name, as specified in the `P4CLIENT` environment variable or its equivalents.	false	string	
owner	The name of the user who owns the workspace. The default is the user who created the workspace. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user	false	string	

Name	Description	Required	Schema	Default
	does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.			
update	The time the workspace specification was last modified.	false	string	
access	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)	false	string	
host	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host. The hostname must be provided exactly as it appears in the output of p4 info when run from that host. This field is meant to prevent	false	string	

Name	Description	Required	Schema	Default
	accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.			
description	A textual description of the workspace. The default text is Created by owner.	false	string	
root	The directory (on the local host) relative to which all the files in the `View:` are specified. The default is the current working directory. The path must be specified in local file system syntax. If you change this setting, you	false	string	

Name	Description	Required	Schema	Default
	must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to map files to multiple drives.			
altRoots	Up to two optional alternate client workspace roots. Perforce applications use the first of the main and alternate roots that match the application's current working directory. Use the p4 info command to display the root being used. This enables users to use the same Perforce client workspace specification on multiple platforms, even those with different directory naming conventions. If you are using multiple or alternate workspace roots (the AltRoots: field), you can always tell which	false	[string]	

Name	Description	Required	Schema	Default
	root is in effect by looking at the Client root: reported by p4 info. If you are using a Windows directory in any of your workspace roots, you must specify the Windows directory as your main workspace root and specify your other workspace roots in the AltRoots: field.			
options	A set of seven switches that control particular workspace options. See [Usage Notes](https://www.perforce.com perforce/ perforce/ doc.current/ manuals/ cmdref/ p4_client.html#p4_ for a listing of these options.	false	string	
submitOptions	Options to govern the default behavior of p4 submit. * submitunchanged + All open files (with or without changes) are submitted to the	false	string	

Name	Description	Required	Schema	Default
	depot. This is the default behavior of Perforce.			
	* submitunchanged +reopen + All open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist.			
	* revertunchanged + Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted.			
	* revertunchanged +reopen + Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist.			
	* leaveunchanged + Only those files with content, type, or resolved changes are submitted to			

Name	Description	Required	Schema	Default
	the depot. Any unchanged files are moved to the default changelist. * leaveunchanged +reopen + Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged +reopen, except that no unchanged files are submitted to the depot.			
lineEnd	Configure carriage-return/ linefeed (CR/LF) conversion. See [Usage Notes](https:// www.perforce.com perforce/ doc.current/ manuals/ cmdref/ p4_client.html#p4_ for a listing of these options.	false	string	
stream	Associates the workspace with the specified stream.	false	string	

Name	Description	Required	Schema	Default
	Perforce generates the view for stream-associated workspaces: you cannot modify it manually.			
streamAtChange	A changelist number that sets a back-in-time view of a stream. When StreamAtChange is set, running p4 sync (when called with no arguments) updates the workspace to files at this changelist revision, instead of the head revision. You cannot submit changes (p4 submit returns an error) when StreamAtChange is set, because the workspace view no longer reflects the current stream inheritance. This field is ignored unless the Stream field is also set to a valid stream.	false	string	
serverID	If set, restricts usage of the workspace to the named server. If unset, use is allowed on master server and	false	string	

Name	Description	Required	Schema	Default
	on any replicas of the master other than Edge servers.			
view	Specifies the mappings between files in the depot and files in the workspace. A new view takes effect on the next p4 sync operation.	false	[string]	
changeView	Restricts access to depot paths to a particular point in time. Files specified for the ChangeView field are read-only: they may be opened but not submitted. For example: `//depot/path/…@1000` Revisions of the files in the specified path will not be visible if they were submitted after the specified changelist number. Files matching a ChangeView path may not be submitted.	false	[string]	
type	By default clients are writeable. Specify readonly for short lived clients used in build automation	false	string	

Name	Description	Required	Schema	Default
	scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts. Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.			

ClientsCommand

Name	Description	Required	Schema	Default
client	The client workspace name, as specified in the `P4CLIENT` environment variable or its equivalents.	false	string	
owner	The name of the user who owns the workspace.	false	string	

Name	Description	Required	Schema	Default
	The default is the user who created the workspace. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.			
update	The time the workspace specification was last modified.	false	string	
access	The date and time that the workspace was last used in any way. (Note: Reloading a workspace with p4 reload does not affect the access time.)	false	string	
host	The name of the workstation on which this workspace resides. If included, operations on this client workspace can be run only from this host. If not set, access is allowed from any host.	false	string	

Name	Description	Required	Schema	Default
	The hostname must be provided exactly as it appears in the output of p4 info when run from that host. This field is meant to prevent accidental misuse of client workspaces on the wrong machine. Providing a host name does not guarantee security, because the actual value of the host name can be overridden with the -H option to any p4 command, or with the P4HOST environment variable. For a similar mechanism that does provide security, use the IP address restriction feature of p4 protect.			
description	A textual description of the workspace. The default text is Created by owner.	false	string	
root	The directory (on the local host) relative to which all the files in the **View:** are specified. The	false	string	

Name	Description	Required	Schema	Default
	default is the current working directory. The path must be specified in local file system syntax. If you change this setting, you must physically relocate any files that currently reside there. On Windows client machines, you can specify the root as null to enable you to map files to multiple drives. additionalPropertie			
type	By default clients are writeable. Specify readonly for short lived clients used in build automation scripts. Such clients cannot edit or submit files, but this should not be an issue in build scripts. Using writeable clients in build automation scripts can lead to db.have table fragmentation, which is used to track what files a client has synced. If you are experiencing such issues, use a read-only	false	string	

Name	Description	Required	Schema	Default
	client instead. A readonly client is assigned its own personal db.have database table. The location of this table must first be specified by an administrator with the client.readonly.dir configurable.			
options	A set of seven switches that control particular workspace options. See [Usage Notes](https://www.perforce.com perforce/ doc.current/ manuals/ cmdref/ p4_client.html#p4_ for a listing of these options.	false	string	
submitOptions	Options to govern the default behavior of p4 submit. * submitunchanged + All open files (with or without changes) are submitted to the depot. This is the default behavior of Perforce. * submitunchanged +reopen + All	false	string	

Name	Description	Required	Schema	Default
	open files (with or without changes) are submitted to the depot, and all files are automatically reopened in the default changelist.			
	* revertunchanged + Only those files with content, type, or resolved changes are submitted to the depot. Unchanged files are reverted.			
	* revertunchanged +reopen + Only those files with content, type, or resolved changes are submitted to the depot and reopened in the default changelist. Unchanged files are reverted and not reopened in the default changelist.			
	* leaveunchanged + Only those files with content, type, or resolved changes are submitted to the depot. Any unchanged files are moved to the default changelist.			
	* leaveunchanged +reopen + Only			

Name	Description	Required	Schema	Default
	those files with content, type, or resolved changes are submitted to the depot. Unchanged files are moved to the default changelist, and changed files are reopened in the default changelist. This option is similar to submitunchanged +reopen, except that no unchanged files are submitted to the depot.			
lineEnd	Configure carriage-return/ linefeed (CR/LF) conversion. See [Usage Notes](https:// www.perforce.com perforce/ doc.current/ manuals/ cmdref/ p4_client.html#p4_ for a listing of these options.	false	string	
stream	Associates the workspace with the specified stream. Perforce generates the view for stream-associated workspaces: you cannot modify it manually.	false	string	

CommandResponse

Name	Description	Required	Schema	Default
results	A collection of maps that have various values set by p4d.	false	[object]	

CommandRequest

Name	Description	Required	Schema	Default
object	Don't use this. It's a kludge around a bug in the Java client code generator	false	object	

Counter

Name	Description	Required	Schema	Default
counter	The variable name	false	string	
value	The variable value. Many variables are numerical in nature, which allow you to do atomic increment operations in method calls instead of having to fetch, increment, and save.	false	string	

DepotCommand

Name	Description	Required	Schema	Default
depot	The depot name.	false	string	
owner	The user who owns the depot. By default, this	false	string	

Name	Description	Required	Schema	Default
	is the user who created the depot. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.			
description	A short description of the depot's purpose. Optional.	false	string	
type	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`. A local depot is writable, and is the default depot type. Files reside in the server's root directory and are managed directly by the server. A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy. A remote depot references files	false	string	

Name	Description	Required	Schema	Default
	that reside on other servers, and cannot be written to.			
	The spec depot, if present, automatically archives edited forms.			
	The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.			
	An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-accessed revisions, typically large binaries.			
	A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.			

Name	Description	Required	Schema	Default
address	If the **Type:** is **remote**, the address should be the P4PORT address of the remote server. If the **Type:** is local or spec, this field is ignored.	false	string	
suffix	If the Type: is spec, this field holds an optional suffix for generated paths to objects in the spec depot. The default suffix is .p4s. You do not need a suffix to use the spec depot, but supplying a file extension to your Perforce server's versioned specs enables users of GUI client software to associate Perforce specifications with a preferred text editor. If the Type: is local or remote, this field is ignored.	false	string	
streamDepth	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following	false	string	

Name	Description	Required	Schema	Default
	the depot name of a stream.] This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.			
map	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the … wildcard; for example, a local depot new might have a Map: of `new/…`. If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, `//depot/new/rel2/…`. This directory will be the root of the local representation of the remote depot.	false	string	
specMap	For spec depots, an optional description of which specs	false	[string]	

Name	Description	Required	Schema	Default
	should be saved, expressed as a view.			

DepotsCommand

Name	Description	Required	Schema	Default
depot	The depot name.	false	string	
map	If the Type: is local, spec, or archive, set the map to point to the relative location of the depot subdirectory. The map must contain the … wildcard; for example, a local depot new might have a Map: of `new/…`. If the Type: is remote, set the map to point to a location in the remote depot's physical namespace, for example, `//depot/new/rel2/…`. This directory will be the root of the local representation of the remote depot.	false	string	
type	`local`, `remote`, `spec`, `stream`, `unload`, `archive` or `tangent`. A local depot is writable, and is the default depot	false	string	

Name	Description	Required	Schema	Default
	type. Files reside in the server's root directory and are managed directly by the server.			
	A stream depot is also writable, but contains streams, a type of branch that includes hierarchy and policy.			
	A remote depot references files that reside on other servers, and cannot be written to.			
	The spec depot, if present, automatically archives edited forms.			
	The unload depot, if present, holds infrequently-used metadata (about old client workspaces and labels) that has been unloaded with the p4 unload command.			
	An archive depot is used in conjunction with the p4 archive and p4 restore commands to facilitate offline (or near-line) storage of infrequently-			

Name	Description	Required	Schema	Default
	accessed revisions, typically large binaries. A tangent depot defines a read-only location that holds tangents created by the p4 fetch -t command. The tangent depot named tangent is automatically created by p4 fetch -t if one does not already exist.			
streamDepth	For stream depots, the optional depth to be used for stream paths in the depot, where depth specifies the number of slashes following the depot name of a stream.] This field is used when streams are being created. The default is 1, matching the traditional stream name. You cannot update this value once streams or archive data exist in a depot.	false	string	
description	A short description of the depot's purpose. Optional.	false	string	

DirsCommand

Name	Description	Required	Schema	Default
dir		false	string	

FilesCommand

Name	Description	Required	Schema	Default
depotFile		false	string	
revision		false	string	
change		false	string	
action		false	string	
time		false	string	
type		false	string	

FstatCommand

Name	Description	Required	Schema	Default
depotFile	Depot path to file. For files containing special characters, the filename is displayed containing the ASCII expression of the character's hexadecimal value.	false	string	
movedFile	Name in depot of moved to/from file.	false	string	
shelved	Set to `shelved` if file is shelved.	false	string	
headAction	Action taken at head revision, if in depot. One of: add, edit, delete, branch,	false	string	

Name	Description	Required	Schema	Default
	move/add, move/delete, integrate, import, purge, or archive.			
headChange	Head revision changelist number, if in depot.	false	string	
headRev	Head revision number, if in depot.	false	string	
headType	Head revision type, if in depot.	false	string	
headCharset	Head charset, for unicode files.	false	string	
headTime	Head revision changelist time, if in depot. Time is measured in seconds since 00:00:00 UTC, January 1, 1970.	false	string	
headModTime	Head revision modification time (the time that the file was last modified on the client before submit), if in depot.	false	string	
movedRev	Head revision of moved file.	false	string	
digest	MD5 digest of a file.	false	string	
fileSize	File length in bytes.	false	string	
actionOwner	User who opened the file, if open.	false	string	

Name	Description	Required	Schema	Default
resolved	The number, if any, of resovled integration records.	false	string	
unresolved	The number, if any, of unresolved integration records.	false	string	
reresolvable	The number, if any, of re-resolvable integration records.	false	string	
otherOpens	For each user with the file open, the workspace and user with the open file.	false	[string]	
otherLocks	For each user with the file locked, the workspace and user holding the lock.	false	[string]	
otherActions	For each user with the file open, the action taken.	false	[string]	
otherChanges	The changelist number with this file open.	false	[string]	
resolveActions	Pending integration action.	false	[string]	
resolveBaseFiles	Pending base files.	false	[string]	
resolveBaseRevs	Pending base revision numbers.	false	[string]	
resolveFromFiles	Pending from files.	false	[string]	

Name	Description	Required	Schema	Default
resolveStartFromRe	Pending starting revisions.	false	[string]	
resolveEndFromRe	Pending ending revisions.	false	[string]	

GitFusionRepoId

Name	Description	Required	Schema	Default
id	An identifier for the repository that can be used safely within URL paths.	false	string	
name	The repository name, which can be path-like.	false	string	

GitFusionRepoConfig

Name	Description	Required	Schema	Default
name	The repository name, which can be path-like.	false	string	
description	Repo description returned by the @list command.	false	string	
globalOverrides		false	"GitFusionRepoGlo	ge 95
branches		false	["GitFusionRepoBra	94]

GitFusionRepoBranchConfig

Name	Description	Required	Schema	Default
gitBranchId	Alphanumeric ID for the git branch. *Do not change this value once this repo has been cloned*.	false	string	
gitBranchName	Defines a name specified in a	false	string	

Name	Description	Required	Schema	Default
	local repo for a Git branch. A valid Git branch name. Do not edit this value after you clone the repo.			
view	Defines a Perforce workspace view mapping that maps Perforce depot paths (left side) to Git work tree paths (right side). Correctly formed mapping syntax; must not include any Perforce stream or spec depots, and all depot paths on the right side must match exactly across all branch definitions. You can add and remove only certain types of Perforce branches from this view after you clone the repo.	false	[string]	
stream	Defines a Perforce stream that maps to the Git branch. Provide a stream name using the syntax // streamdepot/ mystream. A Git Fusion branch can be defined as a	false	string	

Name	Description	Required	Schema	Default
	view or a stream but not both. If your branch is defined as stream, it can include only one stream.			
readOnly	Prohibit git pushes that introduce commits to the branch.	false	string	

GitFusionRepoGlobalOverrides

Name	Description	Required	Schema	Default
charset	Defines the default Unicode setting that Git Fusion applies to new repos. This setting is valid only when Git Fusion interacts with a Unicode-enabled Perforce server. (Defaults to `UTF-8`).	false	string	
depotPathRepoCre	Allow Git users to create new repos by pushing/pulling a git url which specifies a Perforce depot path. This is similar to creating a repo from a p4 client.	false	string	
depotPathRepoCre	Restrict which authenticated Git pushers are allowed to create new repos when depot-path-repo-	false	string	

Name	Description	Required	Schema	Default
	creation-enable is enabled.			
changeOwner	Defines whether Git Fusion assigns either the Git commit author or the Git pusher as the owner of a pushed change (submit).	false	string	
enableGitBranchCre	Defines whether Git Fusion creates a new branch of Perforce depot file hierarchy for each copied branch of Git workspace history, including Git task branches as Git Fusion anonymous branches.	false	string	
enableSwarmRevie	Permits branch creation for Swarm reviews, even when enable-git-branch-creation is disabled.	false	string	
enableGitMergeCo	Defines whether Git Fusion copies merge commits and displays them in Perforce as integrations between Perforce branches.	false	string	
enableGitSubmodu	Defines whether Git Fusion allows Git submodules to be pushed to Perforce.	false	string	

Name	Description	Required	Schema	Default
ignoreAuthorPermi	Defines whether Git Fusion evaluates both the author's and pusher's Perforce write permissions during a push or evaluates only the pusher's permissions.	false	string	
preflightCommit	Enables you to trigger pre-flight commit scripts that enforce local policy for Git pushes. This can be especially useful if you have Perforce submit triggers that could reject a push and damage the repository.	false	string	
readPermissionChe	Enables you to require that Git clone, pull, or fetch requests check the Perforce protections table for the puller's read permission on the files being pulled.	false	string	
gitMergeAvoidance	If the Perforce service includes any changelists submitted by Git Fusion 13.2 or earlier, you can prevent unnecessary merge commits by setting this key to the number of the last changelist submitted	false	string	

Name	Description	Required	Schema	Default
	before your site upgraded to a later version of Git Fusion.			
jobLookup	Set the format for entering Perforce jobs in Git commit descriptions so that they are recognized by Git Fusion and appear in Perforce changelists as fixes. By default, job IDs whose string starts with "job" (as in job123456) are passed through to the changelist description and job field. Use this option if you want Git Fusion to recognize additional expressions, such as JIRA issue IDs.	false	string	
depotBranchCreatic	Allow Git users to create new fully-populated depot branches within Perforce.	false	string	
depotBranchCreatic	Restrict the authenticated Git pushers who are allowed to create new fully-populated depot branches, if depotBranchCreatic is enabled.	false	string	

Name	Description	Required	Schema	Default
depotBranchCreatic	Tell Git Fusion where to create new fully-populated depot branches, if depotBranchCreatic is enabled. Default path is `//depot/[repo]/ [git_branch_name]`.	false	string	
depotBranchCreatic	Set how the depot path set in depotBranchCreatic should appear in Git. Enter a Perforce view specification that maps Perforce depot paths (left side) to Git work tree paths (right side). Perforce depot paths are relative to the root set in depotBranchCreatic. The default … … maps every file under the depotBranchCreatic root to Git. Right side paths must match the right side for every other branch already defined within a repo.	false	string	
enableGitFindCopic	When Git reports a copy file action, store that action in Perforce as a p4 integ. Often set in tandem with enableGitFindRena	false	string	

Name	Description	Required	Schema	Default
	No/Off/0%: Do not use Git's copy detection. Treat all possible file copy actions as p4 add actions. 1%-100%: Use Git's copy detection. Value passed to git diff-tree --find-copies=n. Git Fusion also adds --find-copies-harder whenever adding --find-copies.			
enableGitFindRena	When Git reports a rename (also called move) file action, store that in Perforce as a p4 move. Often set in tandem with enableGitFindCopi No/Off/0%: Do not use Git's rename detection. Treat all possible file rename actions as independent p4 delete and p4 add actions. 1%-100%: Use Git's rename detection. Value passed to git diff-tree --find-renames=n.	false	string	
enableStreamImpo	Enables you to convert Perforce stream import paths to Git	false	string	

Name	Description	Required	Schema	Default
	submodules when you clone a Git Fusion repository. If set to Yes, you must also set either httpUrl or sshUrl.			
httpUrl	The URL used by Git to clone a repository from Git Fusion over HTTP. This property is required if you want to use Perforce stream import paths as git submodules and you use HTTP(S).	false	string	
sshUrl	The "URL" used by Git to clone a repository from Git Fusion using SSH. This property is required if you want to use Perforce stream import paths as git submodules and you use SSH.	false	string	
emailCaseSensitivit	Defines whether Git Fusion pays attention to case when matching Git user email addresses to Perforce user account email addresses during the authorization check.	false	string	
authorSource	Defines the source that Git	false	string	

Name	Description	Required	Schema	Default
	Fusion uses to identify the Perforce user associated with a Git push. Defaults to `git-email`. Use any one of the following values: - `git-email`: Use the email address of the Git author to look for a Perforce user account with the same email address. Git Fusion consults the p4gf_usermap file first, and if that fails to produce a match, it scans the Perforce user table. - `git-user`: Use the user.name field in the Git commit. This is the part of the author field before the email address. - `git-email-account`: Use the account portion of the Git author's email address. If the Git author's email value is `samwise@the_shire` Git Fusion uses the Perforce account samwise.			

Name	Description	Required	Schema	Default
	You can also tell Git Fusion to iterate through multiple source types until it finds a matching Perforce account. Specify the source types in order of precedence, separated by commas. For example: git-user, git-email-account, git-email.			
limitSpaceMb	Natural number representing the number of megabytes of disk space that can be consumed by any single repo. This value does not include the spaced consumed on the Perforce server.	false	string	
limitCommitsRecei	Natural number representing the maximum number of commits allowed in a single push.	false	string	
limitFilesReceived	Natural number representing the maximum number of files allowed in a single push.	false	string	
limitMegabytesRec	Natural number representing the maximum number of megabytes	false	string	

Name	Description	Required	Schema	Default
	allowed in a single push.			

GroupCommand

Name	Description	Required	Schema	Default
group	The name of the group, as entered on the command line.	false	string	
maxResults	The maximum number of results that members of this group can access from the service from a single command. The default value is **unset**.	false	string	
maxScanRows	The maximum number of rows that members of this group can scan from the service from a single command. The default value is **unset**.	false	string	
maxLockTime	The maximum length of time (in milliseconds) that any one operation can lock any database table when scanning data. The default value is **unset**.	false	string	
maxOpenFiles	The maximum number of files that a member of a group can open using a single command.	false	string	

Name	Description	Required	Schema	Default
timeout	The duration (in seconds) of the validity of a session ticket created by p4 login. The default value is 43,200 seconds (12 hours). To create a ticket that does not expire, set the Timeout: field to `unlimited`.	false	string	
passwordTimeout	The length of time (in seconds) for which passwords for users in this group remain valid. To disable password aging, use a value of unset.	false	string	
ldapConfig	The LDAP configuration to use when populating the group's user list from an LDAP query.	false	string	
ldapSearchQuery	The LDAP query used to identify the members of the group.	false	string	
ldapUserAttribute	The LDAP attribute that represents the user's username.	false	string	
subgroups	Names of other Perforce groups. To add all users in a previously defined group to the group	false	[string]	

Name	Description	Required	Schema	Default
	you're presently working with, include the group name in the Subgroups: field of the p4 group form. Note that user and group names occupy separate namespaces, and thus, groups and users can have the same names. Every member of any previously defined group you list in the Subgroups: field will be a member of the group you're now defining.			
owners	Names of other Perforce users. Group owners without super access are permitted to administer this group, provided that they use the -a option. Group owners are not necessarily members of a group; if a group owner is to be a member of the group, the userid must also be added to the Users: field.	false	[string]	

Name	Description	Required	Schema	Default
	The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.			
users	The Perforce usernames of the group members.	false	[string]	

GroupsCommand

Name	Description	Required	Schema	Default
user		false	string	
group		false	string	
isSubGroup		false	string	
isOwner		false	string	
isUser		false	string	
maxResults		false	string	
maxScanRows		false	string	
maxLockTime		false	string	
maxOpenFiles		false	string	
timeout		false	string	
passTimeout		false	string	

HWSStatus

Name	Description	Required	Schema	Default
status	When "OK" the server should be considered to be operating normally	false	string	
version	The version of Helix Web Services server.	false	string	

JobCommand

Name	Description	Required	Schema	Default
Job	The job name.	false	string	

JobsCommand

Name	Description	Required	Schema	Default
Job	The job name.	false	string	

LabelsCommand

Name	Description	Required	Schema	Default
label	The label name.	false	string	
update	The date the label specification was last modified.	false	string	
access	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload	false	string	

Name	Description	Required	Schema	Default
	does not affect the access time.)			
owner	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.	false	string	
options	Options to control behavior and storage location of labels. - locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync. - autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload	false	string	

Name	Description	Required	Schema	Default
	is set, it is stored in the unload depot. This option is ignored for automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels.			
description	An optional description of the label's purpose.	false	string	

LabelCommand

Name	Description	Required	Schema	Default
label	The label name.	false	string	
owner	The label's owner. By default, the user who created the label. Only the owner of a label can update which files are tagged with the label. The specified owner does not have to be a Perforce user. You might want to use an arbitrary name if the user does not yet exist, or if you have deleted the user and need a placeholder until you can assign the spec to a new user.	false	string	

Name	Description	Required	Schema	Default
update	The date the label specification was last modified.	false	string	
access	The date and time the label was last accessed, either by running p4 labelsync on the label, or by otherwise referring to a file with the label revision specifier @label. (Note: Reloading a label with p4 reload does not affect the access time.)	false	string	
description	An optional description of the label's purpose.	false	string	
options	Options to control behavior and storage location of labels. - locked or unlocked: If the label is locked, the list of files tagged with the label cannot be changed with p4 labelsync. - autoreload or noautoreload. For static labels, if noautoreload is set, the label is stored in db.label, and if autoreload is set, it is stored in the unload depot. This option is ignored for	false	string	

Name	Description	Required	Schema	Default
	automatic labels. Storing labels in the unload depot can improve performance on sites that make extremely heavy use of labels.			
revision	An optional revision specification for an automatic label. If you use the # character to specify a revision number, you must use quotes around it in order to ensure that the # is parsed as a revision specifier, and not as a comment field in the form.	false	string	
view	A list of depot files that can be tagged with this label. No files are actually tagged until **p4 labelsync** is invoked. Unlike client views or branch views, which map one set of files to another, label views consist of a simple list of depot files.	false	[string]	
serverID	If set, restricts usage of the label to the named	false	string	

Name	Description	Required	Schema	Default
	server. If unset, this label may be used on any server.			

Location

Name	Description	Required	Schema	Default
depotPath	An absolute depot path specification.	false	string	
depot		false	"DepotsCommand"	
dir		false	"DirsCommand" o	
file		false	"FilesCommand" o	
fstat		false	"FstatCommand" o	
content	If this location indicates a single file, this can be set with the Base64-encoded content of the file.	false	string	

LoginRequest

Name	Description	Required	Schema	Default
user	Usually the Perforce username	false	string	
password		false	string	
serverLogins		false	["ServerLoginRequ	

ServerLoginRequest

Name	Description	Required	Schema	Default
id	The server's ID	false	string	
user		false	string	

Name	Description	Required	Schema	Default
password		false	string	

LoginResponse

Name	Description	Required	Schema	Default
ticket		false	string	

P4dConfigId

Name	Description	Required	Schema	Default
id	A simple string identifier (alphanumeric characters only, please)	false	string	
name	A display string, not guaranteed to be unique	false	string	
description	A simple textual description, for potential selection by clients.	false	string	

Protections

Name	Description	Required	Schema	Default
protections	Each item in the protections array is a line in the protections table, and is split into five columns. 1. Access level or mode. One of the access levels list, read, open, write, admin, super, review; or one of the rights =read, =open, =write, and =branch,	false	[string]	

Name	Description	Required	Schema	Default
	2. Either **user** or **group**, to indicate what's identified by this entry.			
	3. The group name or user name. To grant permission to all users, use a wildcard with just an asterix symbol.			
	4. The IP address of the client host.			
	5. The depot file path, which can contain wildcards. To exclude this mapping from the permission set, use a dash **-** as the first character of this value.			
	IPv6 addresses and IPv4 addresses are also supported. You can use the * wildcard to refer to all IP addresses, but only when you are not using CIDR notation.			
	If you use the * wildcard with an IPv6 address, you must enclose the entire IPv6 address in square brackets. For example, [2001:db8:1:2:*] is equivalent to			

Name	Description	Required	Schema	Default
	[2001:db8:1:2::]/64. Best practice is to use CIDR notation, surround IPv6 addresses with brackets, and to avoid the * wildcard.			
	How the system forms host addresses depends on the setting of the dm.proxy.protects variable. By default, this variable is set to 1. This means that if the client host uses some intermediary (proxy, broker, replica) to access the server, the proxy- prefix is prepended to the client host address to indicate that the connection is not direct. If you specify proxy-* for the Host field, that will affect all connections made via proxies, brokers, and replicas. A value like proxy-10.0.0.5 identifies a client machine with an IP address of 10.0.0.5 that is connected to the server through an intermediary.			

Name	Description	Required	Schema	Default
	Setting the dm.proxy.protects variable to 0, removes the proxy- prefix and allows you to write a single set of protection entries that apply both to directly- connected clients as well as to those that connect via an intermediary. This is more convenient but less secure if it matters that a connection is made using an intermediary. If you use this setting, all intermediaries must be at release 2012.1 or higher.			

ServersCommand

Name	Description	Required	Schema	Default
serverID	A unique identifier for this server. This must match the contents of the server's `server.id` file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.	false	string	

Name	Description	Required	Schema	Default
type	Server executable type. One of the following: `server`, `proxy`, `broker`, `identifier`, `admin`. Each type may offer one or more services, defined in the `services` property.	false	string	
services	The `server` type server provides the following services: - standard - a standard Perforce server - replica - a read-only replica server - commit-server - central server in distributed installation - edge-server - node in distributed installation - forwarding-replica - a replica configured to forward commands that involve database writes to a master server - build-server - a replica that supports build automation and build farm integration - P4AUTH - a server that provides	false	string	

Name	Description	Required	Schema	Default
	authentication - P4CHANGE - a server that provides change numbering - depot-master - commit-server with automated failover - depot-standby - standby replica of the depot-master - workspace-server - node in a cluster installation - standby - read-only replica server that uses p4 journalcopy - forwarding-standby - forwarding replica server that uses p4 journalcopy			
	The **proxy** type server provides a p4p caching proxy.			
	The **broker** type server provides the following services:			
	- broker - a p4broker process - workspace-router - routing broker for a cluster			
	The services field for the **identifier** type server specifies the existence of the cluster, and			

Name	Description	Required	Schema	Default
	has the value `cluster`. The name of the cluster is then drawn from the ServerID field. The `admin` type server provides the following services: - hxca-server - the admin server for a Helix cluster. - zookeeper-server - ZooKeeper server for a cluster			
name	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.	false	string	
address	The P4PORT used by this server.	false	string	
description	An optional description for this server.	false	string	
user	The service user name used by the server.	false	string	

ServerCommand

Name	Description	Required	Schema	Default
serverID	A unique identifier for this server. This must match the contents	false	string	

Name	Description	Required	Schema	Default
	of the server's **server.id** file as defined by the p4 serverid command. If the server type is identifier, the server id specifies the name of the cluster.			
type	Server executable type. One of the following: **server**, **proxy**, **broker**, **identifier**, **admin**. Each type may offer one or more services, defined in the **services** property.	false	string	
services	The **server** type server provides the following services: - standard - a standard Perforce server - replica - a read-only replica server - commit-server - central server in distributed installation - edge-server - node in distributed installation - forwarding-replica - a replica configured to forward commands that	false	string	

Name	Description	Required	Schema	Default
	involve database writes to a master server - build-server - a replica that supports build automation and build farm integration - P4AUTH - a server that provides authentication - P4CHANGE - a server that provides change numbering - depot-master - commit-server with automated failover - depot-standby - standby replica of the depot-master - workspace-server - node in a cluster installation - standby - read-only replica server that uses p4 journalcopy - forwarding-standby - forwarding replica server that uses p4 journalcopy The **proxy** type server provides a p4p caching proxy. The **broker** type server provides the following services: - broker - a p4broker process			

Name	Description	Required	Schema	Default
	- workspace-router - routing broker for a cluster The services field for the `identifier` type server specifies the existence of the cluster, and has the value `cluster`. The name of the cluster is then drawn from the ServerID field. The `admin` type server provides the following services: - hxca-server - the admin server for a Helix cluster. - zookeeper-server - ZooKeeper server for a cluster			
name	The P4NAME associated with this server. You can leave this blank or you can set it to the same value as the serverid.	false	string	
address	The P4PORT used by this server.	false	string	
externalAddress	For an edge server, this optional field specifies the external address used for	false	string	

Name	Description	Required	Schema	Default
	connections to a commit server. This field must be set for the edge server to enable parallel submits in a federated environment.			
description	An optional description for this server.	false	string	
user	The service user name used by the server.	false	string	
clientDataFilter	For a replica server, this optional field can contain one or more patterns describing how active client workspace metadata is to be filtered. Active client workspace data includes have lists, working records, and pending resolves. To include client data, use the syntax: `//client-pattern/…` To exclude client data, use the syntax: `-//client-pattern/…` All patterns are specified in client syntax.	false	string	

Name	Description	Required	Schema	Default
revisionDataFilter	For a replica server, this optional field can contain one or more patterns describing how submitted revision metadata is to be filtered. Submitted revision data includes revision records, integration records, label contents, and the files listed in submitted changelists. To include depot data, use the syntax: To exclude depot data, use the syntax: -//depot/ pattern/… All patterns are specified in depot syntax.	false	string	
archiveDataFilter	For a replica server, this optional field can contain one or more patterns describing the policy for automatically scheduling the replication of file content. If this field is present, only those files described by the pattern are automatically	false	string	

Name	Description	Required	Schema	Default
	transferred to the replica; other files are not transferred until they are referenced by a replica command that needs the file content.			
	Files specified in the ArchiveDataFilter: field are transferred to the replica regardless of whether any users of the replica have made requests for their content.			
	To automatically transfer files on submit, use the syntax: **//depot/ pattern/…**			
	To exclude files from automatic transfer, use the syntax: **-//depot/ pattern/…**			
	All patterns are specified in depot syntax.			
distributedConfig	For an edge or commit server, this optional field, which is displayed only when you use the -l or -c option, shows configuration settings for this server.	false	string	

Name	Description	Required	Schema	Default
	`-l` flag shows the current configuration. `-c-` flag shows current configuration values, recommended default values for fields that are not set, or unset for fields that are not set and do not have default values. If this field is present when invoked with -c, the configuration commands in this field are run on the current server using the scope of the server specified in the serverID field.			

StreamCommand

Name	Description	Required	Schema	Default
stream	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form `//depotname/streamname`.	false	string	
update	The date the stream specification was last modified.	false	string	
access	The date and time that the stream specification was	false	string	

Name	Description	Required	Schema	Default
	last accessed by any Perforce command.			
owner	The Perforce user or group who owns the stream. The default is the user who created the stream.	false	string	
name	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname portion of the stream path.	false	string	
parent	The parent of this stream. Must be none if the stream's `Type:` is mainline, otherwise must be set to an existing stream identifier of the form `//depotname/streamname`.	false	string	
type	The stream's type determines the expected flow of change. Valid stream types are `mainline`, `virtual`, `development`, and `release`. - `mainline`: The mainline stream is the parent of all streams in the stream depot. Every stream	false	string	

Name	Description	Required	Schema	Default
	depot must have at least one mainline stream. - **virtual**: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream. - **release**: More stable than the mainline. Release streams copy from the parent and merge to the parent. - **development**: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent.			

Name	Description	Required	Schema	Default
	- `task`: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.			
description	Description of the stream.	false	string	
options	Settings that configure stream behavior as follows: - `[un]locked`: Enable/disable other users' ability to edit or delete the stream. If locked,	false	string	

Name	Description	Required	Schema	Default
	the stream specification cannot be deleted, and only its owner can modify it. The default is unlocked. - [all,owner]submit: Specifies whether all users or only the owner of the stream can submit changes to the stream. The default is allsubmit. If the Owner: of a stream marked ownersubmit is a group, all users who are members of that group can submit changes to the stream. - [no]toparent: Specifies whether integrations from the stream to its parent are expected. The default is toparent. - [no]fromparent: Specifies whether integrations to the stream from its parent are expected. The default is fromparent for mainline and development streams, and nofromparent for release streams.			

Name	Description	Required	Schema	Default
	-mergeany,mergedow Specifies whether the merge flow is restricted or whether merge is permitted from any other stream. For example, the mergeany option would allow a merge from a child to a parent with no warnings. A virtual stream must have its flow options set to notoparent and nofromparent. Flow options are ignored for mainline streams.			
paths	Paths define how files are incorporated into the stream structure. Specify paths using the following format: path_type view_path [depot_path] where path_type is a single keyword, view_path is a file path with no leading slashes, and the optional depot_path is a file path beginning with //. The default path is share …	false	[string]	

Name	Description	Required	Schema	Default
	Valid path types are: - `share` `view_path`: Specified files can be synced, submitted, and integrated to and from the parent stream. - `isolate` `view_path`: Specified files can be synced and submitted, but cannot be integrated to and from the parent stream. - `import` `view_path` `[depot_path]`: Specified files can be synced, but cannot be submitted or integrated to and from the parent stream. The `view_path` is mapped as in the parent stream's view, or to an (optional) `depot_path`. The `depot_path` may include a changelist specifier. That stream's client workspaces will be limited to seeing revisions at that change or lower within that depot path. For			

Name	Description	Required	Schema	Default
	example, you can specify a depot path like this: **// depot/import/… @1000**. Revisions from changelists greater than 1000 will be automatically hidden from most commands. The changelist limits in effect for a given stream workspace are displayed in a read-only client workspace specification field called ChangeView. - **import+ view_path [depot_path]**: Functions like a standard import path, enabling you to map a path from outside the stream depot to your stream, but unlike a standard import path, you can submit changes to the files in an import + path. - **exclude view_path**: Specified files cannot be synced, submitted or integrated to and from the parent stream. By default, streams			

Name	Description	Required	Schema	Default
	inherit their structure from the parent stream (except mainlines, which have no parent). Paths are inherited by child stream views; a child stream's path can downgrade the inherited view, but not upgrade it. (For example, a child stream can downgrade a shared path to an isolated path, but if the parent stream defines a path as isolated, its child cannot restore full access by specifying the path as shared.) Note that the `depot_path` is relevant only when the path_type is `import` or `import +`.			
remapped	Reassigns the location of workspace files. To specify the source path and its location in the workspace, use the following syntax: `view_path_1 view_path_2` where `view_path_1` and `view_path_2` are Perforce	false	[string]	

Name	Description	Required	Schema	Default
	view paths (omit leading slashes and leading or embedded wildcards; terminal wildcards are fine). For example, to ensure that files are synced to the local ProjectX folder, remap as follows: … `projectX/`… Line ordering in the Remapped: field is significant: if more than one line remaps the same files, the later line takes precedence. Remappings are inherited by child streams and the workspaces associated with them.			
ignored	A list of file or directory names to be ignored in client views. For example: `` ` /tmp # ignores files named "tmp" /tmp/ … # ignores directories named "tmp" .tmp # ignores file names ending in .tmp ` `` Lines in the `Ignored:` field	false	[string]	

Name	Description	Required	Schema	Default
	can appear in any order. Ignored files and directories are inherited by child stream client views.			

StreamsCommand

Name	Description	Required	Schema	Default
stream	Specifies the stream's name (permanent identifier) and its path in the stream depot, in the form `//depotname/streamname`.	false	string	
update	The date the stream specification was last modified.	false	string	
access	The date and time that the stream specification was last accessed by any Perforce command.	false	string	
owner	The Perforce user or group who owns the stream. The default is the user who created the stream.	false	string	
name	Display name of the stream. Unlike the Stream: field, this field can be modified. Defaults to the streamname	false	string	

Name	Description	Required	Schema	Default
	portion of the stream path.			
parent	The parent of this stream. Must be none if the stream's `Type:` is mainline, otherwise must be set to an existing stream identifier of the form `// depotname/ streamname`.	false	string	
type	The stream's type determines the expected flow of change. Valid stream types are `mainline`, `virtual`, `development`, and `release`. - `mainline`: The mainline stream is the parent of all streams in the stream depot. Every stream depot must have at least one mainline stream. - `virtual`: Virtual streams allow merging and copying between parent and child streams without storing local data. Data is passed through to the destination (a non-virtual stream) after applying restrictions on	false	string	

Name	Description	Required	Schema	Default
	the scope of files defined in the virtual stream's view. Because virtual streams do not have files in their depot namespace, it is impossible to import a virtual stream.			
	- `release`: More stable than the mainline. Release streams copy from the parent and merge to the parent.			
	- `development`: Less stable than the mainline. Development streams expect to merge from parent streams and copy to the parent.			
	- `task`: Task streams are lightweight short-lived branches that are useful for bug fixing or new features that only modify a small subset of the branch data. Because branched (copied) files are tracked in a set of shadow tables which are later removed, repository metadata is kept to a minimum			

Name	Description	Required	Schema	Default
	when using this type of stream. Workspaces associated with task streams see all branched data, but only modified and promoted data is visible to users with access to the stream's namespace. The default is stream type is development.			
description	Description of the stream.	false	string	
options	Settings that configure stream behavior as follows: - `[un]locked`: Enable/disable other users' ability to edit or delete the stream. If locked, the stream specification cannot be deleted, and only its owner can modify it. The default is unlocked. - `` `[all ``	owner]submit`: Specifies whether all users or only the owner of the stream can submit changes to the stream. The default is allsubmit. If the Owner: of a stream marked ownersubmit is a group, all users who are members of that group can submit changes to the stream.  - `[no]toparent`: Specifies whether integrations from the stream to its parent are expected. The default is toparent.  - `[no]fromparent`: Specifies whether integrations	mergedown`: Specifies whether the merge flow is restricted or whether merge is permitted from any other stream. For example, the mergeany option would allow a merge from a child to a parent with no warnings. A virtual stream must have its flow options set to notoparent and nofromparent. Flow options are ignored for mainline streams.	false

Name	Description	Required	Schema	Default
	to the stream from its parent are expected. The default is fromparent for mainline and development streams, and nofromparent for release streams. - `mergeany			

Triggers

Name	Description	Required	Schema	Default
triggers	A list of trigger definitions. A trigger definition contains four fields that specify the name of the trigger, the type of event that should trigger the execution of the script, the paths that should be affected by the trigger, the location of the script, and other trigger type-dependent information. When the condition specified in a trigger definition is satisfied, the associated script or program is executed. Example: `trig1` `change-submit //`	false	[string]	

Name	Description	Required	Schema	Default
	depot/dir/… "/ usr/bin/s1.pl %changelist%" See the Helix Versioning Engine Administrator Guide for more details on trigger definitions.			

UserCommand

Name	Description	Required	Schema	Default
user	The Perforce username.	false	string	
type	Type of user: standard, operator, or service. Once you set the type, you cannot change it.	false	string	
authMethod	One of the following: perforce or ldap. Specifying perforce enables authentication using Perforce's internal db.user table or by way of an authentication trigger. This is the default unless it is overridden with the auth.default.method configurable. Specifying ldap enables authentication	false	string	

Name	Description	Required	Schema	Default
	against AD/LDAP servers specified by the currently active LDAP configurations.			
email	The user's email address. By default, this is user@client.	false	string	
update	The date and time this specification was last updated.	false	string	
access	The date and time this user last ran a Perforce command.	false	string	
fullName	The user's full name.	false	string	
jobView	Jobs matching this jobview appear on any changelists created by this user. Jobs that are fixed by the changelist should be left in the changelist when it's submitted with p4 submit; other jobs should be deleted from the form before submission.	false	string	
password	The user's password.	false	string	
passwordChange	The date and time of the user's last password change. If the user has no	false	string	

Name	Description	Required	Schema	Default
	password, this field is blank.			
reviews	A list of files the user would like to review. This field can include exclusionary mappings.	false	[string]	

UsersCommand

Name	Description	Required	Schema	Default
user	The Perforce username.	false	string	
type	Type of user: standard, operator, or service. Once you set the type, you cannot change it.	false	string	
email	The user's email address. By default, this is user@client.	false	string	
update	The date and time this specification was last updated.	false	string	
access	The date and time this user last ran a Perforce command.	false	string	
fullName	The user's full name.	false	string	
hasPassword	If 'enabled', the password has been set on the user.	false	string	

Appendices

Appendix A: Third Party Software Licenses

Name	License	Homepage
commons-codec	"Apache License, 2.0" on page 595	https://commons.apache.org/proper/commons-codec/
commons-logging	"Apache License, 2.0" on page 595	https://commons.apache.org/proper/commons-logging/
google-http-client, google-http-client-gson, google-http-client-jackson2	"Apache License, 2.0" on page 595	https://developers.google.com/api-client-library/java/google-http-java-client/
gson	"Apache License, 2.0" on page 595	https://github.com/google/gson
httpclient	"Apache License, 2.0" on page 595	https://hc.apache.org/httpcomponents-client-ga/
httpcore	"Apache License, 2.0" on page 595	https://hc.apache.org/httpcomponents-client-ga/
jackson-core	"Apache License, 2.0" on page 595	http://wiki.fasterxml.com/JacksonHome
javax.servlet	"COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0" on page 601	
java-configparser	"Apache License, 2.0" on page 595	https://github.com/ASzc/java-configparser
jetty-http, jetty-io, jetty-sercurity, jetty-server, jetty-servlet, jetty-util, jetty-webapp, jetty-xml, websocket-api, websocket-client, websocket-common, websocket-server, websocket-servlet	"Apache License, 2.0" on page 595	https://eclipse.org/jetty/
jna	"Apache License, 2.0" on page 595	https://github.com/java-native-access/jna
jsr-305	"BSD License, 3-clause" on page 599	https://code.google.com/p/jsr-305/

Name	License	Homepage
jzlib	"BSD License, 3-clause" on page 599	http://www.jcraft.com/jzlib/
log4j	"Apache License, 2.0" on page 595	http://logging.apache.org/log4j
slf4j-api, slf4j-log4j12	"MIT Licenses" on page 600	http://www.slf4j.org/
spark-core	"Apache License, 2.0" on page 595	http://sparkjava.com/
yamlbeans	"MIT Licenses" on page 600	https://github.com/EsotericSoftware/yamlbeans

Apache License, 2.0

```
                     Apache License
               Version 2.0, January 2004
            http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
```

copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

(a) You must give any other recipients of the Work or
    Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices
    stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
    that You distribute, all copyright, patent, trademark, and
    attribution notices from the Source form of the Work,
    excluding those notices that do not pertain to any part of
    the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
    distribution, then any Derivative Works that You distribute must
    include a readable copy of the attribution notices contained
    within such NOTICE file, excluding those notices that do not
    pertain to any part of the Derivative Works, in at least one
    of the following places: within a NOTICE text file distributed
    as part of the Derivative Works; within the Source form or
    documentation, if provided along with the Derivative Works; or,
    within a display generated by the Derivative Works, if and
    wherever such third-party notices normally appear. The contents
    of the NOTICE file are for informational purposes only and
    do not modify the License. You may add Your own attribution
    notices within Derivative Works that You distribute, alongside
    or as an addendum to the NOTICE text from the Work, provided
    that such additional attribution notices cannot be construed
    as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A

PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

BSD License, 3-clause

jzlib

JZlib 0.0.* were released under the GNU LGPL license. Later, we have switched
over to a BSD-style license.

```
-------------------------------------------------------------------------------
Copyright (c) 2000-2011 ymnk, JCraft,Inc. All rights reserved.
```

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

  1. Redistributions of source code must retain the above copyright notice,
     this list of conditions and the following disclaimer.

  2. Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in
     the documentation and/or other materials provided with the distribution.

  3. The names of the authors may not be used to endorse or promote products
     derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL JCRAFT,
INC. OR ANY CONTRIBUTORS TO THIS SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

jsr305

```
Copyright (c) 2007-2009, JSR305 expert group
All rights reserved.
```

http://www.opensource.org/licenses/bsd-license.php

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

   * Redistributions of source code must retain the above copyright notice,
     this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above copyright notice,
     this list of conditions and the following disclaimer in the documentation
     and/or other materials provided with the distribution.
   * Neither the name of the JSR305 expert group nor the names of its
     contributors may be used to endorse or promote products derived from
     this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

MIT Licenses

slf4j

Copyright (c) 2004-2013 QOS.ch
All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,   FITNESS   FOR   A   PARTICULAR   PURPOSE   AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

yamlbeans

Copyright (c) 2008 Nathan Sweet, Copyright (c) 2006 Ola Bini

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0

COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0

1. Definitions.

   1.1. Contributor. means each individual or entity that creates or contributes to the creation of Modifications.

   1.2. Contributor Version. means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.

   1.3. Covered Software. means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.

   1.4. Executable. means the Covered Software in any form other than Source Code.

   1.5. Initial Developer. means the individual or entity that first makes Original Software available under this License.

   1.6. Larger Work. means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

   1.7. License. means this document.

   1.8. Licensable. means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

   1.9. Modifications. means the Source Code and Executable form of any of the following:

       A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;

       B. Any new file that contains any part of the Original Software or previous Modification; or

       C. Any new file that is contributed or otherwise made available under the terms of this License.

   1.10. Original Software. means the Source Code and Executable form of computer software code that is originally released under this License.

   1.11. Patent Claims. means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

   1.12. Source Code. means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

   1.13. You. (or .Your.) means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, .You. includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, .control. means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants.

2.1. The Initial Developer Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).

(c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.

2.2. Contributor Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Availability of Source Code.

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

3.2. Modifications.
The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

3.3. Required Notices.
You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

3.4. Application of Additional Terms.
You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients. rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.5. Distribution of Executable Versions.
You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipient.s rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.6. Larger Works.
You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

4. Versions of the License.

4.1. New Versions.
Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

4.2. Effect of New Versions.

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

4.3. Modified Versions.
When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

5. DISCLAIMER OF WARRANTY.

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN .AS IS. BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. TERMINATION.

6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as .Participant.) alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

6.3. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.

7. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR

LOST PROFITS, LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER
COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH
DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY
RESULTING FROM SUCH PARTY.S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME
JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO
THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

8. U.S. GOVERNMENT END USERS.

   The Covered Software is a .commercial item,. as that term is defined in 48 C.F.R. 2.101 (Oct.
1995), consisting of .commercial computer software. (as that term is defined at 48 C.F.R. ?
252.227-7014(a)(1)) and .commercial computer software documentation. as such terms are used in
48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through
227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those
rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other
FAR, DFAR, or other clause or provision that addresses Government rights in computer software under
this License.

9. MISCELLANEOUS.

   This License represents the complete agreement concerning subject matter hereof. If any provision
of this License is held to be unenforceable, such provision shall be reformed only to the extent
necessary to make it enforceable. This License shall be governed by the law of the jurisdiction
specified in a notice contained within the Original Software (except to the extent applicable
law, if any, provides otherwise), excluding such jurisdiction.s conflict-of-law provisions. Any
litigation relating to this License shall be subject to the jurisdiction of the courts located
in the jurisdiction and venue specified in a notice contained within the Original Software, with
the losing party responsible for costs, including, without limitation, court costs and reasonable
attorneys. fees and expenses. The application of the United Nations Convention on Contracts for the
International Sale of Goods is expressly excluded. Any law or regulation which provides that the
language of a contract shall be construed against the drafter shall not apply to this License. You
agree that You alone are responsible for compliance with the United States export administration
regulations (and the export control laws and regulation of any other countries) when You use,
distribute or otherwise make available any Covered Software.

10. RESPONSIBILITY FOR CLAIMS.

   As between Initial Developer and the Contributors, each party is responsible for claims and
damages arising, directly or indirectly, out of its utilization of rights under this License and
You agree to work with Initial Developer and Contributors to distribute such responsibility on
an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of
liability.

   NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)

   The code released under the CDDL shall be governed by the laws of the State of California
(excluding conflict-of-law provisions). Any litigation relating to this License shall be subject to
the jurisdiction of the Federal Courts of the Northern District of California and the state courts
of the State of California, with venue lying in Santa Clara County, California.

The GNU General Public License (GPL) Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA
02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the

Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

  a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

  b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

  c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

  a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/ donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

One line to give the program's name and a brief idea of what it does.

Copyright (C)

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

"CLASSPATH" EXCEPTION TO THE GPL VERSION 2

Certain source files distributed by Sun Microsystems, Inc. are subject to the following clarification and special exception to the GPL Version 2, but only where Sun has expressly included in the particular source file's header the words

"Sun designates this particular file as subject to the "Classpath" exception as provided by Sun in the License file that accompanied this code."

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License Version 2 cover the whole combination.

As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module.? An independent module is a module which is not derived from or based on this library.? If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so.? If you do not wish to do so, delete this exception statement from your version.