

Commands Reference

- [ActiveMQ](#): An ActiveMQ deployment
 - [assertServiceIsDown](#): checks if the service is down
 - [assertServiceIsUp](#): checks if the service is running
 - [Docs-Generate](#): generates all defined docs
 - [Prepare](#): builds the deployment objects
 - [startService](#): starts the service
 - [stopService](#): stops the service
 - [Update](#): run the update cycle
- [ActiveMQZip](#): Standard platform zip format package
 - [extract](#): Extract the package archive.
- [AntBuilder](#): A simple Builder to interface with Ant
 - [generateHudsonAntInstallation](#): Generate a Hudson Ant installation xml fragment
 - [generateHudsonConfig](#): Generate a Hudson job (project) config.xml file
 - [generateHudsonJdkInstallation](#): Generate a Hudson JDK installation xml fragment
 - [generateProject](#): Generate a CruiseControl project definition for inclusion in config.xml
 - [runBuildScript](#): runs the build script
 - [writeBuildProperties](#): Write out properties file for use in Hudson
- [AntZip](#): An Apache Ant Distribution Package
 - [assertPackageIsInstalled](#): Confirm the package process is installed.
 - [extract](#): Extract the package archive.
 - [finish](#): finishes the installation
- [Apache](#): Manages an Apache HTTP server
 - [assertServiceIsDown](#): checks if service is down
 - [assertServiceIsUp](#): checks if service is running
 - [Docs-Generate](#): generates all defined docs
 - [Prepare](#): builds the deployment objects
 - [startService](#): starts the service
 - [stopService](#): stops the service
- [ApacheSystemService](#): Apache System Service Module
 - [Docs-Generate](#): Generate the Apache configuration file
- [BatBuilder](#): A simple Builder to interface with Windows batch script based builds

- [Build](#): Run the build cycle.
- [generateProject](#): Generate a CruiseControl project definition for inclusion in config.xml
- [Import](#): Import a set of packages.
- [runBuildScript](#): runs the build script
- [BitTorrentBuilder](#): BitTorrent torrent file builder based on the CreateTorrent utility by Daniel Etzold
 - [Build](#): Run the build cycle.
 - [runBuildScript](#): runs the build script
- [BitTorrentClient](#): BitTorrent client management service
 - [Deploy](#): Run the service deployment cycle, stopping installing package dependencies, configuring and the starting it.
- [BitTorrentFile](#): BitTorrent "torrent" file
- [BitTorrentTracker](#): BitTorrent tracker server
- [CTierInstallerZip](#): ControlTier Installer Package zip
- [ClientInstaller](#): client installer module
 - [configureClient](#): configure a new controltier client
 - [Deploy](#): automates installation and project configuration for a new controltier client
- [ConfigZip](#): Configuration zip archive
- [ContentZip](#): A generic Content zip package
- [ContinuousIntegration](#): Continuous integration server
- [CruiseControl](#): Cruise Control continuous integration facility
 - [assertServiceIsDown](#): checks if process is down
 - [assertServiceIsUp](#): checks if process is running
 - [Docs-Generate](#): generates all defined docs
 - [startService](#): start the service process
 - [stopService](#): stops the service process
- [CruiseControlService](#): Run Cruise Control as a Windows service using Tanuki Software's Java Service Wrapper facility
 - [Docs-Generate](#): Generate the Java Service Wrapper and Cruise Control configuration files
- [CruiseControlZip](#): Zip archive of Cruise Control
 - [assertPackageIsInstalled](#): Confirm the package process is installed.
 - [extract](#): Extract the package archive.
- [DukesBankProjectBuilder](#): Builds projects used to manage instances of the Dukes Bank application
 - [Prepare](#): Prepares the demo

- [scmImport](#): Imports sources to SCM repository
- [scmInitialize](#): Initializes SCM repository
- [ElementsProjectBuilder](#): Builds and manages projects that use the Elements modules library
- [FireDaemonService](#): Service integration of FireDaemon Technologie's Windows Service wrapper
 - [Add](#): Add the FireDaemon Windows service if it is not already installed.
 - [addService](#): Add the FireDaemon Windows service to the system
 - [assertServiceIsDown](#): Check whether the FireDaemon Windows service is down
 - [assertServiceIsInstalled](#): Check whether the FireDaemon Windows service is installed
 - [assertServiceIsNotInstalled](#): Check whether the FireDaemon Windows service is not installed
 - [assertServiceIsUp](#): Check whether the FireDaemon Windows service is up
 - [Deploy](#): Run the service deployment cycle, stopping, unconfiguring, installing package dependencies, configuring and the starting the service.
 - [Remove](#): Remove the FireDaemon Windows service if it is installed.
 - [removeService](#): Remove the FireDaemon Windows service from the system
 - [startService](#): Start the FireDaemon Windows service
 - [stopService](#): Stop the FireDaemon Windows service
 - [Update](#): Run the service deployment cycle, same as Deploy
- [HsqldbRdb](#): A Hypersonic SQL database service
 - [assertServiceIsDown](#): Check whether the Hypersonic SQL database server instance is down
 - [assertServiceIsUp](#): Check whether the Hypersonic SQL database server instance is up
 - [Deploy](#): Run the service deployment cycle, stopping installing package dependencies, configuring and the starting it.
 - [Docs-Generate](#): Generate the server properties file
 - [startService](#): Start the Hypersonic SQL database server instance
 - [stopService](#): Stop the Hypersonic SQL database server instance
- [HsqldbRdbDmp](#): Hypersonic SQL database dump package
- [HsqldbRdbExportBuilder](#): Hypersonic SQL database export builder
 - [Build](#): run the build cycle
 - [runBuildScript](#): runs the build script
- [HsqldbRdbSchema](#): A Hypersonic SQL database schema
- [HsqldbZip](#): Zip archive of the Hypersonic SQL Database
 - [extract](#): Extract the package archive.
- [Hudson](#): Hudson continuous integration server
 - [assertServiceIsDown](#): checks if process is down

- [assertServiceIsUp](#): checks if process is running
- [Docs-Generate](#): Generates Hudson configuration files based on builders
- [startService](#): start the service process
- [stopService](#): stops the service process
- [HudsonPlugin](#): Hudson Plugin (.hpi) file
- [HudsonWar](#): Hudson continuous integration server war package
- [ImageMagickTgz](#): ImageMagick tgz package
 - [extract](#): Extract the package archive.
- [JBossAntBuilder](#): JBoss application Ant builder
- [JBossEar](#): ATG Java enterprise application archive package
 - [extract](#): Extract the package archive.
 - [finish](#): finishes the package installation
- [JBossServer](#): A JBoss application server
 - [assertServiceIsDown](#): Check whether JBoss is down
 - [assertServiceIsUp](#): Check whether JBoss is up
 - [Configure](#): Run the configuration cycle for the deployment.
 - [Docs-Baseline](#): Experimental: Creates a baseline for the configuration documents required for jboss operation
 - [Docs-Generate](#): creates directories required for jboss operation
 - [Docs-Register](#): Experimental: registers configuration documents required for jboss operation
 - [Docs-Verify](#): Experimental: Verifies the configuration documents required for jboss operation
 - [killService](#): kills the jboss server process
 - [packages-code-install](#): installs all the code package dependencies
 - [packages-config-install](#): installs all the Configuration package dependencies
 - [Start](#): Conditionally starts the JBoss Server
 - [startService](#): asynchronously executes the jboss start command
 - [startServiceWrapper](#): wraps the jboss start procedure and waits for start event
 - [Stop](#): Conditionally stops the JBoss Server
 - [stopService](#): executes the jboss stop command
 - [stopServiceWrapper](#): wraps the jboss stop procedure and waits for stop event
 - [waitForStartEvent](#): waits for a startup even of an asynchronously started jboss service
 - [waitForStopEvent](#): waits for a stop event of a jboss service
- [JBossZip](#): A JBoss Distribution Package
 - [assertPackageIsInstalled](#): Confirm the package process is installed.
 - [extract](#): Extract the package archive.
- [JavaBin](#): A unix based self extracting java platform package

- [assertPackageIsInstalled](#): Confirm the package process is installed.
- [extract](#): Extract the package archive.
- [JavaServiceWrapper](#): Windows platform integration of Tanuki Software's Java Wrapper Service facility
 - [Add](#): Add the Java Service Wrapper Windows service if it is not already installed.
 - [addService](#): Add the Java Service Wrapper Windows service to the system
 - [assertServiceIsDown](#): Check whether the Java Service Wrapper Windows service is down
 - [assertServiceIsInstalled](#): Check whether the Java Service Wrapper Windows service is installed
 - [assertServiceIsNotInstalled](#): Check whether the Java Service Wrapper Windows service is not installed
 - [assertServiceIsUp](#): Check whether the Java Service Wrapper Windows service is up
 - [Deploy](#): Run the service deployment cycle, stopping, unconfiguring, installing package dependencies, installing the dependent Service, and configuring and the starting the service. .
 - [dispatchServiceConfigure](#): Dispatch the Configure command to a dependent service object
 - [dispatchServiceInstall](#): Dispatch the Install command to a dependent service object
 - [dispatchServicePackagesInstall](#): Dispatch the Packages-Install command to a dependent service object.
 - [Docs-Generate](#): Generate the server properties file
 - [Remove](#): Remove the Java Service Wrapper Windows service if it is installed.
 - [removeService](#): Remove the Java Service Wrapper Windows service from the system
 - [Service-Install](#): Install a dependent Service object without starting it directly
 - [startService](#): Start the Java Service Wrapper Windows service
 - [stopService](#): Stop the Java Service Wrapper Windows service
- [JavaServiceWrapperZip](#): Zip archive of Tanuki Software's Java Service Wrapper facility
- [JavaZip](#): java zip package
- [JtdsJar](#): A platform jar package
- [MavenBuilder](#): A simple Builder to interface with Maven
 - [generateHudsonConfig](#): Generate a Hudson job (project) config.xml file
 - [generateHudsonJdkInstallation](#): Generate a Hudson JDK installation xml fragment
 - [generateProject](#): Generate a CruiseControl project definition for inclusion in config.xml
 - [installPlugin](#): Install the ControlTier Maven plugin in the local Maven repository
 - [runBuildScript](#): runs the build script
 - [writeBuildProperties](#): Write out properties file for use in Hudson
- [MavenZip](#): Zip archive of Maven

- [extract](#): Extract the package archive.
- [Mule](#): Mule Enterprise Service Bus
 - [assertServiceIsUp](#): checks if process is running
 - [startService](#): start the service process
 - [stopService](#): stops the service process
- [MuleUserJar](#): Jar archive of the user contributed Mule assets destined for "\$MULE_HOME/lib/user"
- [MuleZip](#): Zip archive of the Mule distribution
 - [finish](#): finishes the package installation
- [MysqlRdb](#): A MySQL database instance
 - [assertServiceIsDown](#): checks if the service is down
 - [assertServiceIsUp](#): checks if the service is running
 - [Docs-Generate](#): generates all defined docs
 - [installDb](#): installs the database
 - [Prepare](#): creates the db and prepares it for use
 - [startService](#): starts the service
 - [stopService](#): stops the service
- [MysqlSchema](#): A mysql schema
 - [createSchema](#): creates the schema
 - [grantSchema](#): grants access to the schema
- [OpenLDAP](#): OpenLDAP Lightweight Directory Access Protocol Server
 - [assertServiceIsDown](#): Check whether the OpenLDAP "slapd" process is down
 - [assertServiceIsUp](#): Check whether the OpenLDAP "slapd" process is up
 - [Docs-Generate](#): Generate the OpenLDAP "slapd" configuration files
 - [startService](#): Start the OpenLDAP "slapd" process
 - [stopService](#): Stop the OpenLDAP "slapd" process
- [PackedWar](#): Packed (unexploded) web archive package
- [PlatformJar](#): A platform jar package
 - [extract](#): Extract the package archive.
 - [finish](#): finish the extract of jar
- [PlatformZip](#): Standard platform zip format package
 - [finish](#): finishes the package installation
- [PostgresRdb](#): PostgreSQL server system service module
 - [assertServiceIsUp](#): Check whether the specified system service is up
 - [Configure](#): Run the configuration cycle for the deployment.
 - [Deploy](#): Run the service deployment cycle, stopping installing package dependencies, configuring and the starting it.

- [Docs-Generate](#): Generate the PostgreSQL server configuration files
- [InitDb](#): Invoke the PostgreSQL server database initialization program (initdb)
- [PostgresRdbSchema](#): A PostgreSQL database schema
- [PuppetClient](#): Puppet client management service
- [PuppetMaster](#): The Puppet Master service
- [Rdb](#): A relational database service
 - [assertServiceIsUp](#): checks if the service is up
 - [Deploy](#): Runs the Rdb update process
 - [Schemas-ExportDmp](#): Calls each RdbSchema resource to export its database schema to a dump package
 - [Schemas-ImportDmp](#): Calls each RdbSchema resource to import its dump file package
- [RdbSchema](#): Represents a database schema
 - [export](#): Exports the schema
 - [ExportDmp](#): exports and uploads a dump
 - [import](#): Imports the schema
 - [ImportDmp](#): Imports the dmp file
 - [registerDmp](#): registers the dump file as an object
 - [Status](#): Gets the status of the service
 - [upSchema](#): Checks schema is available in the designated instance
- [RpmBuilder](#): An RPM package builder.
 - [Build](#): Run the build cycle.
 - [createPackage](#): Create package(s)
 - [generateManifest](#): Generate list of files to package.
 - [generateProject](#): Generate a CruiseControl project definition for inclusion in config.xml
 - [setBuildstamp](#): Optionally automatically generate the buildstamp, updating the model if required
- [SolidcoreController](#): Models the Solidcore controller's management of a set of nodes
- [SolidcoreHost](#): Solidcore S3 System Host Service
 - [assertServiceIsDown](#): Check whether the host does not have Solidcore protection enabled
 - [assertServiceIsUp](#): Check whether the specified host has Solidcore protection enabled
 - [startService](#): Switch the specified host to end update, starting (enabling) Solidcore protection
 - [stopService](#): Switch the specified host to begin update, stopping (disabling) Solidcore protection
- [Squid](#): Squid caching proxy server

- [assertServiceIsDown](#): checks if the service is down
- [assertServiceIsUp](#): checks if the service is running
- [killService](#): kills the squid service
- [purgeObject](#): purges an object from the squid cache
- [startService](#): starts the squid service
- [Stop](#): Conditionally stops the JBoss Server
- [stopService](#): stops the squid service
- [stopServiceWrapper](#): wraps the jboss stop procedure and waits for stop event
- [waitForStopEvent](#): waits for the service to shutdown
- [SystemService](#): Unix/Linux system service module
 - [assertServiceIsDown](#): Check whether the specified system service is down
 - [assertServiceIsUp](#): Check whether the specified system service is up
 - [startService](#): Start the specified system service
 - [stopService](#): Stop the specified system service
- [Tomcat](#): Mediates administration of a Tomcat server and a set of one or more associated contexts. DEPRECATED
 - [Configure](#): Configures all services
 - [Packages-Install](#): install packages for the services
 - [Prepare](#): run the platform preparation cycle
 - [Restart](#): stop and start the site's services
 - [Start](#): starts all services
 - [Status](#): calls status for each service
 - [Stop](#): stops all services
 - [Update](#): run the update cycle
- [TomcatAntBuilder](#): Ant builder with specialized defaults for Tomcat applications
- [TomcatContext](#): Tomcat web application service (extended to support a custom context and an "external" property file)
 - [assertServiceIsDown](#): asserts if service is down
 - [assertServiceIsUp](#): asserts if service is up
 - [Configure](#): Configures the instance
 - [Docs-Generate](#): generates all defined docs
 - [Packages-Install](#): install package dependencies
 - [startService](#): executes the start procedure
 - [stopService](#): executes the stop procedure
 - [War-Packages-Install](#): install war package dependencies
- [TomcatServer](#): Apache Tomcat server deployment
 - [assertServiceIsDown](#): checks if the service is down
 - [assertServiceIsUp](#): checks if the service is running
 - [Docs-Generate](#): Generate associated Documents, and Tomcat configuration files

- based on templates for the specific Tomcat release.
- [startService](#): starts the service
- [stopService](#): stops the service
- [TomcatServerService](#): Run TomcatServer as a Windows service using Tanuki Software's Java Service Wrapper facility
 - [Docs-Generate](#): Generate the Java Service Wrapper and Cruise Control configuration files
- [TomcatSite](#): Centralized management for a set of Tomcat based application server instances and contexts.
 - [Packages-Install](#): Dispatch package installation to the site's resources.
 - [Update](#): Refresh dependencies and then deploy the site.
- [TomcatZip](#): Zip archive of Apache Tomcat
 - [assertPackageIsInstalled](#): Confirm the package process is installed.
 - [extract](#): Extract the package archive.
- [WarUpdater](#): Updater for J2EE web archive based applications
 - [Change-Dependencies](#): Queries model and then switches dependencies
- [WindowsService](#): Automate Windows Service Control
 - [assertServiceIsDown](#): Check whether the specified Windows service is down
 - [assertServiceIsUp](#): Check whether the specified Windows service is up
 - [startService](#): Start the specified Windows service
 - [stopService](#): Stop the specified Windows service
- [ZipBuilder](#): Simple builder to create a Zip file package of the "basedir" (deployment-basedir) in the "targetdir" (deployment-install-root) using the object name as file name and timestamp versioning
 - [Build](#): run the build cycle
 - [runBuildScript](#): runs the build script