

Frequently Asked Questions

Table of contents

1 Questions.....	2
1.1 1. General.....	2
1.2 2. Interoperability.....	6
1.3 3. Security.....	7
1.4 4. Technical.....	8

Questions

1. General

1.1. What is ControlTier?

ControlTier is a software system used to automate the build deployment processes for distributed, multi-tier applications. It was originally developed by [ControlTier Inc.](#) beginning in 2003, and has entered its fourth year of development.

1.2. Licensing and Pricing

ControlTier software is free and open source. See the license information [here](#). There are no hidden "gotchas". Download it. Try it. Use it. If you find value in it and would like to participate, see the [contribution guidelines](#).

1.3. Where do I download ControlTier builds?

Information for getting binaries and source code is available in the [download documentation](#).

1.4. What features are unique to ControlTier?

We believe ControlTier is unique in a number of ways. First of all, ControlTier is a generic platform that can be used by both development and operations groups separately and for their own purposes. Developers can use ControlTier to extend the continuous integration process to include the build, deployment and setup of multi-tier applications in their own integration test environments. Operations can use ControlTier to manage and validate the deployment of large scale application environments, control the startup and shutdown of multi-tier components, or use ControlTier as an inventory and configuration database of the environment. The best part is that you are using the same reliable and repeatable automation in every environment and ControlTier's object-oriented framework makes your automation far more reusable and easier to maintain than relying just on traditional ad-hoc scripting methods.

Development and operations groups can also collaborate via the ControlTier platform to improve the speed of handoffs between groups and allow for safer delegation of build and deployment tasks to less skilled staff .

Some specific features that make ControlTier attractive include:

- full featured tools for versioned modeling

- graphical automation workflow builder
- automation that is portable across environments
- integrates the configuration *and* procedural model
- logical management and network transparency
- fine grain access control (role based)
- centralized logging and activity or compliance reporting
- ctld more...

1.5. What does ControlTier do out of the box?

Since ControlTier is a platform containing both a set of applications and a framework, its capabilities are deep in some areas and broad in others. ControlTier's applications are full featured to support the model and automation development process. Once you have plugged in your environment information and scripts it generates and distributes modules where you can execute your automation. There are many features within the Workbench application providing model development, reporting, logging, archiving, and versioning.

Where ControlTier is broad in its capabilities is its use and support of procedural automation that spans your various tools, scripts, package types, and configuration files. ControlTier includes a base automation Type library that provides you the procedural logic you need to manage multi-tier deployments. You can simply plug in your existing tools and scripts, or you can extend those Types to any unique situation that may arise in your environment. At a framework level, ControlTier includes a large number of Ant tasks that provide a range of functionality needed to develop and manage complex processes.

What you are probably really wondering is, does ControlTier support versions of specific packages like JBoss, Tomcat, Apache, Weblogic, Websphere, Oracle, Mysql, etc? ControlTier does have modules for some of these platforms and more are being added all the time. The truth is though, different users are interested managing some things one way while another user wants to do it in quite another. Therefore, one module does not always fit all. We seek to establish the common denominators as building blocks in various automation libraries.

1.6. How does ControlTier compare to Tivoli, Clearcase, etc?

Because ControlTier is a generic and somewhat comprehensive platform people have asked how does it compare to other development and operations management tools. First of all, let's point out what ControlTier is not:

- **ControlTier is not a version control system.** It is not meant to replace your favorite SCM tool (e.g., CVS, SVN, P4, etc.). ControlTier versions model data and helps you maintain a repository of versioned release artifacts but ControlTier is not a place where

you would want to maintain your source code.

- **ControlTier is not a bare metal provisioning tool** (e.g., Kickstart, Jumpstart, etc.). Bare metal provisioning tools typically support net booting, manage OS profiles and are good for "stamping" out server operating system configurations. ControlTier can be used to manage the configuration and execution of a bare metal provisioning tool but is not meant to take its place. After a provisioning tool has installed a base OS it might also install the ControlTier agent so the new machine can receive release changes.
- **ControlTier is not a project workflow management tool** (e.g., MS project, kintana, remedy, bugzilla, etc.). ControlTier is not meant to be used as an approval-based change control system to manage the people and their activities. The role of these "people management" tools are still central to coordinating the release activity. ControlTier would come into play to actually *execute* the change. ControlTier does augment the greater change control system by introducing its own level of visibility into how the changes are executed, where, how (at a technical level), when, securely, etc.
- **ControlTier is not made to replace software build tools.** (e.g., ant, maven, cruisecontrol, buildforge, etc.). ControlTier is often used to interface build tools so that they can be driven by the ControlTier end-to-end build and deployment process (creating a "Builder"). The line between ControlTier and a java build tool is a bit blurred though because the ControlTier agent embeds Ant (via [CTL](#)). So you can use ControlTier via its tight integration with Ant to build Java software, it's just not the ControlTier primary use.
- **ControlTier is not an enterprise management system** (e.g., Tivoli, HP OpenView, BMC, etc.). These tools are large frameworks for managing many aspects operations groups are concerned with including monitoring, user administration, event correlation, patching, hardware diagnostics, software distribution, etc. ControlTier overlaps with an EMS when it comes to software distribution. Our experience has shown the best way to coordinate the two systems is to use an EMS to manage the system infrastructure and distribute base software platforms that change more or less independently from the normal release activity in the application tier. Where the exact boundary is drawn depends entirely on your organization.

1.7. How do you speed the app lifecycle between DEV, QA, PROD, etc?

From our experience working with customers, the most problematic aspect of managing change from development on through to production is coordinating the technical procedures across teams. There is a lot of knowledge that must be passed for both major and minor releases including:

- environment details
- configuration files and settings
- scripts
- file locations and permissions

- dependencies and sequencing

We have found that each team might even have their own tools and processes to accomplish the changes in their environments. When you add up all the miscommunication, manual errors, wheel reinvention and inconsistencies, it is no wonder releases are so error-prone, risky and slow.

The solution is for each group to share a release platform, to institute an architecture for how the change execution will be developed and managed, and then share a common set of automation modules so that each group is executing exactly the same procedure.

Using ControlTier, typically the development group bootstraps this process, creating an application model of the integrated system and develops and tests the automation in their first integration environment. This setup is then saved as a `pattern` in ControlTier and that pattern can be sent to downstream teams where they install the pattern in their environment and execute the automation modules.

1.8. Can ControlTier provision whole environments?

ControlTier can play a couple of different roles in provisioning an entire environment. ControlTier can work in conjunction with an OS provisioning system as explained in separate [faq entry](#). ControlTier can also be used to install, configure and setup all the needed software packages needed to support the application tier (the typical arrangement).

1.9. How do I get started? config data, files, scripts, builds, etc.

The general subject of "getting started" is taken up in the user documentation available from the Help section of the Workbench interface. But, the general methodology is to pick a single process within the app lifecycle, start small and incrementally, develop the configuration and control models. Beginning users often believe they need to account for every resource, configuration setting, dependency and procedure on the outset because they see value in documenting these things. While this is a great long term goal, we have found that taking a pragmatic stance that concentrates on useful automation goals is the most efficient method.

If you are new to ControlTier it is highly recommended that you start with the [CTL ProjectBuilder Tutorial](#)

1.10. Can ControlTier be used for managing disaster recovery?

ControlTier should be part of the disaster recovery strategy since it is ideal for documenting and executing the installation, setup and startup of multi-tier application systems. Once the application environment has been documented as a ControlTier project or pattern, this

definition along with the ControlTier repository can be replicated to a ControlTier server in a disaster recovery site.

1.11. I live and breathe this problem. Why don't I solve this myself?

It is easy to under estimate what it takes to solve this problem. If you are curious to know what goes into building this kind of solution take a look at our [solution recipe](#).

ControlTier's object-oriented framework also makes it easier to maintain and extend any automation you may create and has built-in collaboration, logging, reporting, and security features that are essential to any enterprises operations.

2. Interoperability

2.1. Is ControlTier cross-platform?

The ControlTier server and agent components are both [Java](#) based and hence cross platform in that regard. ControlTier uses [CTL](#) as the core of the agent, the generated code is [Ant](#)-based, which also provides cross platform scripting.

ControlTier server is essentially a set of Java-based servlets that use a REST style architecture for external integration. The server has been developed and tested on [Tomcat](#) but other servlet containers should also work.

ControlTier agent is a standalone java application that bundles Ant. Theoretically, wherever Ant can run, the ControlTier agent can run.

2.2. What operating systems does ControlTier run on?

ControlTier server and agent have been tested on various versions of Redhat (9, fedora, AS), Solaris 8,9, Mac OSX, Windows (XP, Server2003, 2000).

2.3. Can you integrate with my favorite version control system?

The primary interface to integrating with version control systems is currently via [Ant tasks](#). These include: Clearcase, Continuum/Synergy, CVS, SVN, PVCS, Perforce, StarTeam, and MS SourceSafe.

2.4. Can you integrate with JMX?

Depending on your needs, [JMX4Ant](#) might meet your requirements.

3. Security

3.1. How is ControlTier secure?

ControlTier was designed with security in mind and security is managed at various levels. We believe by using ControlTier to manage your release activity, you will significantly improve security around the process.

- **Authentication:** Workbench users must authenticate to access information managed in Workbench. Workbench uses a form-based login mechanism that has session expiration control. User information is stored in an LDAP database. Commander users must also authenticate to execute commands, as well as, authenticate to the ControlTier server to access information or content in the repository.
- **Authorization:** Authorization is managed via security roles defined in LDAP. A security role is a privilege granted to a group of users allowing a particular Workbench action. Workbench requires users to have a login ID to authenticate to the server, and to determine the user's role, which provides authorization to access different functionality. See the Security Roles section in the Workbench documentation for more info. Whenever a user executes a command via the Commander client, the request passes through an authorization stage, where the user request is matched up against the access control list. Part of the authorization process includes a user role lookup from an LDAP repository. Commander uses an authorization configuration file that can grant access based on deployment context, user group, module and command, as well as time of day.
- **Auditing:** All commands, software failures, application actions and errors are logged centrally by the Workbench application and written to a file local to the server. The Workbench application uses Log4j API for all logging.
- **Communication:** Communication between the ControlTier server and clients is done via HTTP. The server can be configured to use SSL. Communication between ControlTier agents is done via SSH.
- **Integrity:** Workbench can be configured to generate modules that are digitally signed. Each file within the module are digitally signed and users can configure to have these modules verified before installation. Project and pattern archives are also digitally signed and verified before installation.

3.2. Can I use SSL?

Yes, ControlTier server and agent can be configured to use SSL. Consult the Security section in the respective manuals.

3.3. Can I encrypt sensitive data in the model?

Typically, this question is asked in regards to passwords used to access services, databases, etc. Workbench supports the ability encrypt Setting values. Users that created the object or have *ctlmin* level access can edit and view the data in clear text. The data is stored as an encrypted string in the RDBMS. When the data is distributed as a view to a ControlTier agent it remains encrypted. An included Ant task must be used to decrypt the data on the target machine.

4. Technical

4.1. How is ControlTier itself deployed?

ControlTier uses a client-server architecture. The ControlTier server is based on several components, a webapp that runs in a servlet container (e.g. Tomcat), a RDBMS (e.g., Mysql), an LDAP server (e.g. OpenLDAP). ControlTier agent is installed on each target machine to which releases and deployments are made. Read more in the [platform overview](#).

It is recommended that you use the [ControlTier Installer](#).

4.2. What is the process for designing and deploying a classic 3-tier app?

The best way to get a feel for this process is to follow the tutorials.

4.3. Can ControlTier handle co-deployment?

Firstly, what do we mean by co-deployment? When we say co-deployment we are referring to the scenario where an administrator would like to install several applications of the same kind on one machine. Co-deployment makes sense when many applications must be run but there is limited hardware to run them. Problems often occur due to collisions of software packages, configuration settings, file paths, etc.

ControlTier enables and supports the co-deployment of applications. First of all because Commander agent is based on [CTL](#), each software deployment is provided its own workspace where instance specific files and directories can be maintained (e.g., logs, bins, var, etc.). Secondly, because ControlTier provides a configuration database developers and administrators can prescribe ahead of time what resources and settings each instance or class of deployment should use. In other words, co-deployment is possible as a by product of ControlTier's ability to achieve environment abstraction for applications.

4.4. What if somebody changes/installs something outside of ControlTier?

For example, ControlTier was used to deploy an instance of Apache and unbeknownst to the

team at large, a rogue hand edited the `httpd.conf` file to change the listening port. We often refer to this scenario as an "uncoordinated change". Changes like these are often the most difficult to diagnose and fix since mis-configurations often don't stop something from running but instead cause it to run differently. If ControlTier was used to manage the deployment of the configuration file, then you can use a provided command to verify it has not been modified.

The Managed-Entity module contains commands called `Doc-Generate` and `Doc-Validate` that work with checksums to verify generated files.

ControlTier does not provide a [Tripwire](#) IDS feature that baselines a server's filesystem state. You can write ControlTier commands that verify the state of specific files and make those preconditions in a command workflow.

4.5. Does ControlTier support undeploy/rollback?

There are two strategies for managing undeploy or rollback using ControlTier: model-driven vs. module-driven.

Model-driven reversion is based on completely model-driven commands and ControlTier's versioned model. By rolling the model back to a previous state, and re-running the deployment, you effectively bring the deployment back to the previous configuration state. The model-driven strategy is the favored approach as it assumes the state of the configuration model has the primary affect over the behavior of commands. In essence, this just means ControlTier encourages a data-driven programming style.

Module-driven reversion relies on rollback logic that is contained in the control module and may or may not be driven from the centrally managed model state. Module-driven reversion is typically only used for custom applications or databases who's internal state must be specially treated. The module-driven rollback strategy typically involves creating versioned directories in the deployments workspace directory, and storing snapshots of the application internal state. The rollback semantics of the module then understand how to roll back to previous snapshots.

4.6. How do you input the dependencies? Can I bulk load data into the model?

Once you have installed the ControlTier stack the very first step is to start defining objects in the model. This can be done via the Workbench web GUI or it can be done programmatically via the included Ant tasks (see `Object-Create/Object-Update`).

4.7. How do I manage the LDAP users for ControlTier?

ControlTier authentication and authorization layers both use LDAP to lookup users and roles. Depending on your preference you may wish to manage the LDAP database through graphical tools or manage it as a text file which is versioned and uploaded to the LDAP server as needed.

If you prefer the graphical approach, we have reported success using [JXplorer](#) for administering OpenLdap. Binding to an ActiveDirectory server has also been successfully reported.

If you prefer the text file approach, we suggest managing an LDIF file and there are a number of script-based utilities to help with that. [LDAP tutorial and support scripts](#)

4.8. Can ControlTier prompt during a command?

ControlTier has built in support for prompting from inside commands. If a user is prompted while using the shell interface, a message is written to stdout and the response read from stdin. If a user is running the graphical console, a dialog is raised.

ControlTier has a unique network-savvy prompting task that should be used by commands that are executing commands in remote modules. In this mode, any command that calls the `prompt` Ant task, will pass the prompt request to a network service that relays the request to the user. This is tunnelled back through the SSH connection with which the remote command was invoked

Prompting can be placed within commands that you develop. Prompts can also be raised during a command workflow failure.

4.9. Database schema change detection?

ControlTier does not have any special support for checking if a database's schema has changed. This kind of check can be added as a command that might call a sql script and then added to a workflow command sequence as a precondition.

4.10. Can ControlTier scan for exception messages in my app server log?

Application servers (i.e., servlet containers, J2EE servers) are notorious for creating the scenario where the server itself starts up fine but the webapp or EJBs that they load and run begin to spew exception messages during their startup. These exception messages might be caused by ports that are not listening, database misconfigurations, etc.

ControlTier provides an Ant task called `FileMonitor` which should be used in a Start workflow sequence that scans the log file for success and failure criteria. This `FileMonitor` runs in its own thread and reads the log file for pre-configured regular expressions. When a

success or failure point is reached, the configured action is then dispatched. We often also use the `WaitFor` task in this process.

4.11. Can you define package/jar dependencies?

ControlTier provides a base class called `Package` which supports dependency relationships. Its subtypes, `jar`, `war`, `rpm`, etc., therefore inherit this capability.

Using ControlTier's correlation query service, you can get back a set of packages all related by their composition hierarchy (i.e., the transitive closure).

4.12. Can I validate/diff config files ?

Generation and validation of files is supported by the framework. There are two base commands in the Managed-Entity module, `Doc-Generate` and `Doc-Verify` that generate and verify a declared document.

4.13. Can commands be run in parallel?

Command workflows can be configured to execute their task list in parallel. The `Workflow` tag has attributes that control the threading and batching. Since commands will execute more or less asynchronously result data must be specially managed. Refer to the `PropertyResults` and `PropertiesQueryTask` for information about how to access and respond to result data.

4.14. Can I insure the right versions are deployed to the right boxes?

Assuming you use ControlTier to deploy the applications and packages to the boxes, then you can use and/or develop verification commands to determine if the state of the box corresponds to the state of the model. We consider this a natural byproduct of using ControlTier.

4.15. How deeply can things be monitored/checked?

The short answer here is, "as deep as you need". Of course, this is a bit misleading since it relies on the tests that you developed and plugged in as commands (usually as part of workflows). Normally, the kinds of tests that you would use or develop include checking if the application services are running (e.g., by checking the process table, accessing a network port, submitting a transaction, etc.).

A best practice that should be followed is to provide a "Status" command for any module that controls a service. Then a top-level module used for centralized logical management can

recursively call the Status command effectively checking each object down the hierarchy.

4.16. What is the format of the ControlTier model?

ControlTier uses a [semantic web](#) approach to managing information about the configuration and control of application environments. Internally, all data is represented in RDF and stored in a relational database. ControlTier uses the excellent [Jena API](#) to access and manipulate the model data. ControlTier uses RDFS along with a handful of its own vocabulary to build ontologies that reflect your environment.

Workbench does provide data export and import of the model data in a number of RDF formats (n-triple, RDF/XML, N3). This makes it possible to use other RDF tools to edit and visualize model data. Of course, care must be used when editing model data outside of Workbench. Only advanced users should consider doing this.