Technical white paper

Flexible Application Modeling with HP CDA

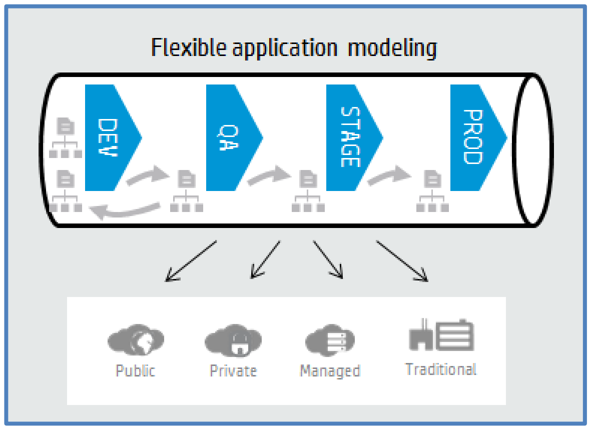
Table of contents

Introduction

HP Continuous Delivery Automation (HP CDA) provides a model-driven approach to automate application deployment in a development and operations (DevOps) environment.

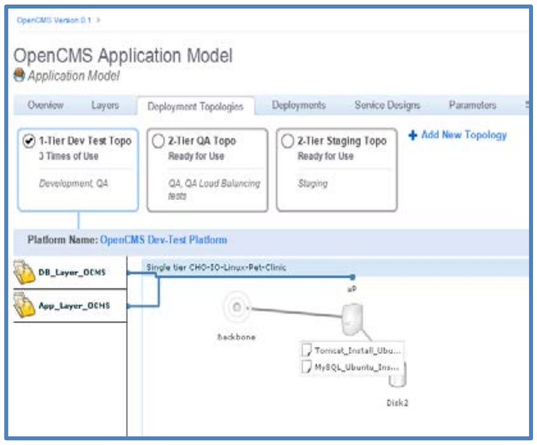
Using this approach, you design applications and platforms separately as *application models* and *platform models*. This results in libraries of such models that can be easily mixed and matched as necessary to deploy the application into a variety of environments that map to your application release lifecycle:

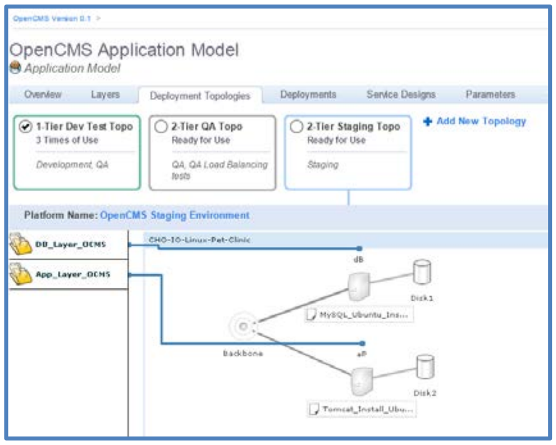
Figure 1. HP Continuous Delivery Automation



For example, a web application, which has been architected to contain separate application and database layers, can be deployed onto a simple, single-tier virtual platform during the development phase of the software development lifecycle, and later deployed onto a multi-tier, multi-server platform during the staging phase of the lifecycle. In both cases, the same application model is used. A similar situation is illustrated in the following screen captures from the HP CDA interface:

Figure 2. Application Deployment to Different Platforms using HP CDA





Development Phase Staging Phase

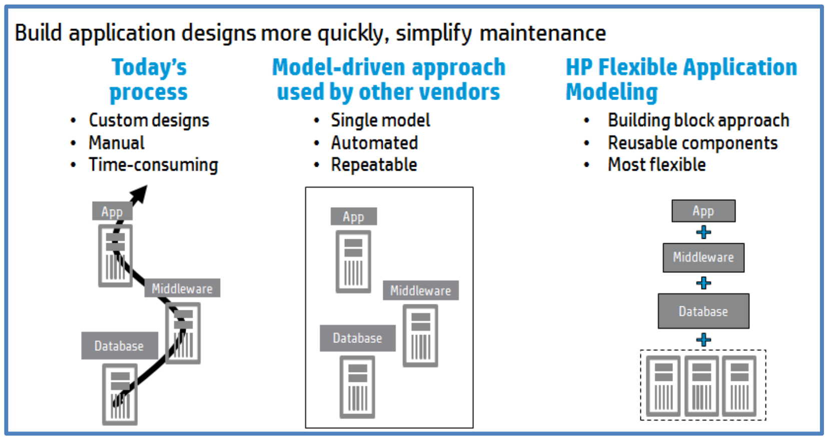
Applications as such can be designed to be platform-agnostic, having instead only a set of specified requirements that can be matched to platform models that have capabilities that meet the requirements of the application. This is the basis of HP CDA's *flexible application modeling* concept.

The HP CDA Difference: Flexible Application Modeling

With flexible application modeling, you design an application that can be deployed and managed in a repeatable fashion–on multiple target platforms. In HP CDA, you create and manage the application model, deploy the application, and manage the deployment, all with full artifact version control. The application model defines and manages modifications, and ensures that whatever is developed and tested is what goes to production.

HP's flexible model-driven process differs from competitors' products by providing flexibility and modularity, versus having a full stack built into a single model which is inflexible, has limited standardization, and results in hundreds of models which can be a maintenance headache.

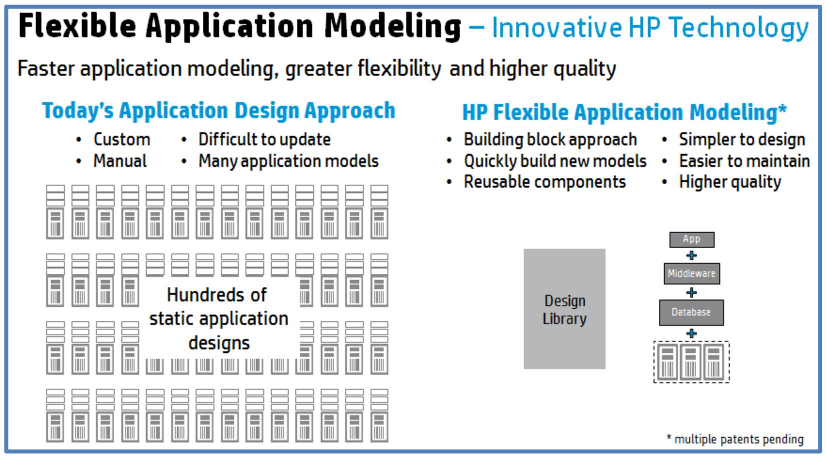
Figure 3. Typical Competitor's Approach


Build application designs more quickly, simplify maintenance
Today's process
• Custom designs
• Manual
• Time-consuming
Model-driven approach used by other vendors
• Single model
• Automated
• Repeatable
HP Flexible Application Modeling
• Building block approach
• Reusable components
• Most flexible
App
Middleware
Database

Benefits of HP's Flexible Application Modeling

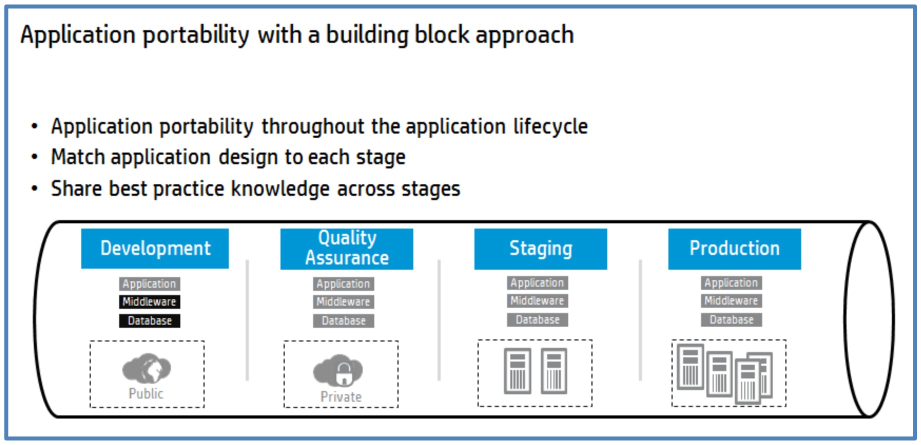
The flexible application modeling approach, coupled with integration with the customer's tools of choice (independence from underlying deployment systems), results in reduced configuration effort (linear complexity versus exponentially increasing complexity), promotes application deployment portability, reduced vendor lock-in and better overall knowledge reuse. Applications are easier to create and modify for a variety of environments. They are also easier to maintain over time, as templates can be created for multiple common applications using CDA's parameter capabilities.

Figure 4. Flexible Application Modeling Benefits


Flexible Application Modeling – Innovative HP Technology
Faster application modeling, greater flexibility and higher quality
Today's Application Design Approach
• Custom
• Manual
• Difficult to update
• Many application models
Hundreds of static application designs
HP Flexible Application Modeling*
• Building block approach
• Quickly build new models
• Reusable components
• Simpler to design
• Easier to maintain
• Higher quality
Design Library
App
Middleware
Database
* multiple patents pending

Application deployment portability and platform agnosticity also simplifies the process of moving an application through the stages of the software development lifecycle.

Figure 5. Application Portability through the Software Development Lifecycle



Application portability with a building block approach

- Application portability throughout the application lifecycle
- Match application design to each stage
- Share best practice knowledge across stages

Since applications are modeled at the functional level, they are not tightly associated with any one technology like Chef, HP SA, or SSH, so they can easily be changed. Any of the following can be swapped out, but the application model remains the same:

- The application deployer (HP or non-HP)
- Source control

Should the customer decide to use a different tool later, the model is not impacted, retaining investment and preventing rework.

Objective of this White Paper

The objective of this white paper is to provide HP CDA users with insights into HP CDA's flexible application modeling capabilities, and provide concrete tips on how to efficiently and effectively create models for your applications.

Fast Track to Application Modeling in HP CDA

So, you have an application for which you want to design an application model using HP CDA. Here are a few simple steps to get you started:

- Identify your application requirements and any dependencies
- Understand (and document) the steps necessary to deploy and install your application, including the execution order of operations
- Start modeling, in a phased approach

As you embark, what are the right questions to ask? Here are a few considerations:

What type of application am I modeling?

For example, is your application an Apache app or a .NET app, or other? This will help you identify the application's requirements, in terms of operating system, hardware, middleware, back-end databases, etc., and any dependencies the application might have. For example, if it is a .NET app, your destination server needs to have .NET installed. Coincidentally, you might have the option to use HP CDA to deploy .NET as platform software, or have it included in ("baked into") the infrastructure template you use.

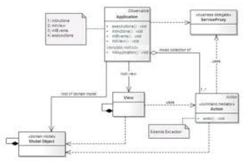
How is my application deployed manually?

Create, or better yet, obtain a "cookbook", if one has been created, which describes in step-by-step fashion how the application is deployed and installed. If scripts are used, you might want to make a high-level list of the steps that are performed. Meet with the persons or teams in your organization who are currently responsible for deploying the application and get their input.

HP CDA can invoke scripts at deploy time, so in most cases you will be able to leverage any scripts you already have.

You are now ready to start modeling your application in HP CDA. Be sure to start small, in a step-by-step fashion. Depending on the complexity of your application and your personal working style, you might choose to construct your application model one step at a time, in a sequential fashion. Then again, you might prefer to create a framework of the overall deployment process, with placeholders for the major operations, and then once you have that in place, go back and start putting in the details. HP CDA works just as well with either approach.

Getting into the Details



Designing Your Application

Once you have a clear understanding of you application's type and its requirements and dependencies, you can start to design (or model) your application in HP CDA. As stated in the previous section, it helps if you have a "cookbook" or other document that describes the step-by-step process used to deploy the application. With this, you can simply map each step to a workflow operation within HP CDA.

Start simple for your first deployment attempt. For example, create a workflow that creates a directory on a destination server and then places a text file in the directory. You can then verify that the file has been properly placed. When creating an application model, go slowly and in small steps. Don't try to create the entire application model, deploy the application, and then figure out where and why it failed.

When creating you first application model, you may want to keep in mind some of HP CDA's unique capabilities provided via flexible application modeling. This approach allows you to define application components independent of infrastructure, and in turn deploy your application very easily and seamlessly in multiple different environments as your application moves through its lifecycle stages.

Specifically, consider using a modular or "building block" approach, and try to design reusability into your model. By creating several small operations, you will be able to easily move them around as needed during the design process. Doing this also provides the ability for you to re-use an application model for other applications that you have. You can take advantage of HP CDA's cloning capabilities to make a clone of an application model that you have created, and then just make any minor modifications necessary to adapt the model for a different application.

Shortcut: download one or more sample application models which are available via HP Live Network (https://hpln.hp.com/group/continuous-delivery-automation), and either use the sample application as an example to refer to as you are creating your application model in HP CDA, or you can even clone the sample application model and modify the cloned model directly for your specific application. You can also download models for several common middleware packages (for example, PHP, Apache, Tomcat, ASP, Jetty, MySQL, SQLite and others), and begin to build out your own middleware software library, which can then be used for other applications that you model in HP CDA. This will help get you up and running faster, with less effort and initial investment required. These models provide you a fast track to getting your first application modeled and deployed.

Using HP CDA's parameterization capabilities, you can create an application model which is essentially like a generic template which can then serve for deploying dozens of applications, provided they have relatively similar requirements and dependencies. For example, if you have 20 or 30 .NET applications, this approach would be applicable. You can then couple this with HP CDA's ability to call out to other workflows within your application model, such that you can start a middleware service or run a script at any time during the deployment, and then return back to your application model. This feature – workflow delegation – allows you to maximize your flexibility of deployments, in turn catering towards even the most complex application deployments, and at the same time begin to build out libraries of workflows that can be used over-and-over again by dozens or even hundreds of applications.

Following this approach, you can not only build common models that can be leveraged across many applications, but also greatly minimize your application maintenance efforts as your applications evolve. Take advantage of HP CDA's component updates capabilities which impact all designs. For example, if you have a hundred applications, currently using Tomcat7, and a new version of Tomcat becomes available, you can make the change in *one* central location in HP CDA, and this change will automatically be cascaded down to all the consuming applications.

Finally, consider making your application or platform model available to others, in a self-service portal context. CDA allows models to be published to HP's Cloud Services Automation application, which provides a user friendly, self-service portal interface from which users of any skill level can easily select Iaas provisioning and application deployment services.

Application Modeling Tips and Tricks

HP CDA's flexible application modeling process has been designed to mimic your current processes, and should feel very intuitive even as you first start out. You simply begin to define the operations, one by one, and modify or move them around as you need.

By starting with a simple deployment scenario, for example an application model that creates a directory on a destination server and then places a file into that directory, you will be able to verify several additional aspects "under the hood", such as connectivity to your target servers, potential deployer handshake issues, and other configuration settings.

Once you have a simple deployment under your belt, you can then quickly branch out and start to explore some of the more advanced capabilities, and exploit HP CDA's re-use capabilities even further. For example, do you already have scripts that perform some portion of the deployment or post-deployment/configuration tasks? If so, you can use HP CDA to invoke these scripts directly, allowing you to use the tools of your choice and protect investments that you may have made already. Or if you'd like to invoke content from another application as part of your deployment process – for example, executing scripts, packages or policies from HP Server Automation, or running flows from HP Operations Orchestration or HP Database and Middleware Automation – you can instruct HP CDA to make these call-outs directly at any point during the deployment process.

When creating your first few application models, proceed slowly and in small steps. Don't try to create the entire model at once and then try to deploy it, as this will make it more difficult to debug issues in case of deployment failure. Rather, create two or three operations that comprise the initial part of your deployment, execute these to make sure they are successful, then go onto to create the next set of operations for your deployment, verify those work as expected, and so on.

Layers and Workflows

With HP CDA's flexible application modeling capabilities, you can design and model your application as you want, and in a variety of different ways to suit your needs. There is no one right approach – you can experiment and find which approach works best for you and your applications. Indeed, depending upon the application type, technology, complexity, etc., the approach you take for application modeling may vary from application to application.

A few guidelines, in terms of best practices, will get you off to a good start.

First, determine how many Layers you want to create for your application model. Typical applications may be comprised of two or three layers (for example, an Application Layer, a DB Layer and potentially also a Web Layer). Correctly identifying your application layers will facilitate in the deployment process later on, as you will see when you create your application's Deployment Topologies and map your application layers to your various platform tiers. This will also pave the way for you to exploit HP CDA's application portability capabilities (via independent modeling of the application and infrastructure), which allows you to map any combination of your applications to environments in a simple and flexible drag-and-drop fashion.

Some of your applications will be multi-layered, as described above, while others may be defined as a single layer only (typically appropriate for more simple applications).

How you order your Layers can be important. For example, if one layer has dependencies on another layer (for example, Layer 2 operations need Layer 1 operations to have already been executed). If you need, you can always easily re-order the layers you've defined, instructing HP CDA which Layer execution must occur first.

If after creating and ordering your Layers, you determine that you need greater granularity, you can always create additional layers. A very basic example of this might be where a portion of the deployment needs to occur first from the App Layer, then a portion of the deployment needs to occur from the DB layer, and then the remaining portion of the App Layer needs to be performed, and finally the remaining portion of the DB Layer. You can easily accommodate this scenario by creating four layers, instead of two:

APP LAYER - Part 1 → DB LAYER - Part 1 → APP LAYER - Part 2 → DB LAYER - Part 2

Once you've defined your Layers, you can apply the same approach to the specific operations that make up your deployment process. You will also determine which workflows you want to use for your application. Typical application models will make use of the Deploy, Redeploy, and Undeploy workflows.

Design your application model so that you can take advantage of HP CDA's Redeploy capabilities. There are several ways to accomplish this, but basically, you will want to include the operations of your deployment *that need to be performed only once in the Deploy workflow, and include the operations of your deployment that need to be performed each time you want to redeploy the application in the Redeploy workflow. For example, if as part of your deployment you need to perform certain setup steps like making temp directories and setting environment variables, these should be defined under the Deploy workflow, whereas operations like placing dynamic, rapidly changing artifacts onto the target servers should be defined under the Redeploy workflow. In this way, you can quickly update an application that has already been deployed by performing a partial, selective deployment without having to perform a full deployment each time you just need to push out a new version of an artifact.*

Applications may also make use of additional workflows such as Start and Stop, as well as defining any Custom workflows that may be required.

Helpful Tips

Create your own Custom Workflows to align with your internal business processes. Custom workflows can be easily created in HP CDA and can be used for operations such as Pause, Restart, Backup, and any other relevant operations.

Just as the ordering of Layers is important, how you order the operations within your layers can also be very important. Again, this is beneficial especially where there may be dependencies between operations. Coupling HP CDA's ability to call workflows between layers provides even greater flexibility in this regard, for example, if operation #3 in App1 Layer needs to have operations #2 and #5 from the DB1 layer run first.

Helpful Tips

If you want additional control over the applications you deploy, you can utilize HP CDA's Start and Stop workflows. In this way, you can deploy an application to your target server(s), and leave them in an un-started (or not running) state until you specifically select to Start the application. In other instances, you may want to instruct HP CDA to deploy and start all associated services automatically, so that your application will be up and running immediately after it has been deployed, in a simple "one step deploy" fashion.

Once you've created an application model, you should take time to define both Undeploy and Backout operations. Undeploy is a default workflow in HP CDA which allows you to specify operations that will be performed to reverse the application deployment process, and essentially restore the target server(s) to a known good, pre-deployment, state. It is important to specify these operations because failing to do so will potentially trigger additional errors (false negatives, for example) the next time you try to deploy the application to the same target server(s). This might prevent you from performing any successful subsequent deployments of the application, as the target server(s) have been left in a partially deployed, or "polluted" state. Making use of the Undeploy workflow is also very helpful when debugging your application deployments.

Helpful Tips

If you plan to deploy more than one application to a target server, and the deployed applications use any shared resources (for example, the same backend database), you should be careful how you instruct HP CDA to undeploy the application. For example, in this instance you will not want to remove MS-SQL altogether, but rather just delete the specific database and/or database tables used by the application that you want to undeploy. In this way, the shared resources that are being used by the other applications that are still deployed and running on the target server(s) will be left in place.

Backout operations can be defined so that selected operations will be performed in the event an application deployment fails in mid-stream. This can be specified for each operation listed in your Deploy and Redeploy workflows. Defining Backout operations is very important, as in the event of a failed deployment, you will not be able to select the Undeploy option for any failed deployments.

Helpful Tips

If you are working with a complex application that may require many operations as part of the undeployment and you are deploying to virtual machines, you may want to specify a single Undeploy operation which will revert your target server(s) to a specific snapshot.

Helpful Tips

A practical way for you to make offline modifications to your application or platform models is to use HP CDA's export option. This will create an XML definition of your model which you can then modify and import back into HP CDA. You can use this feature to export models from one instance of HP CDA to another as well.

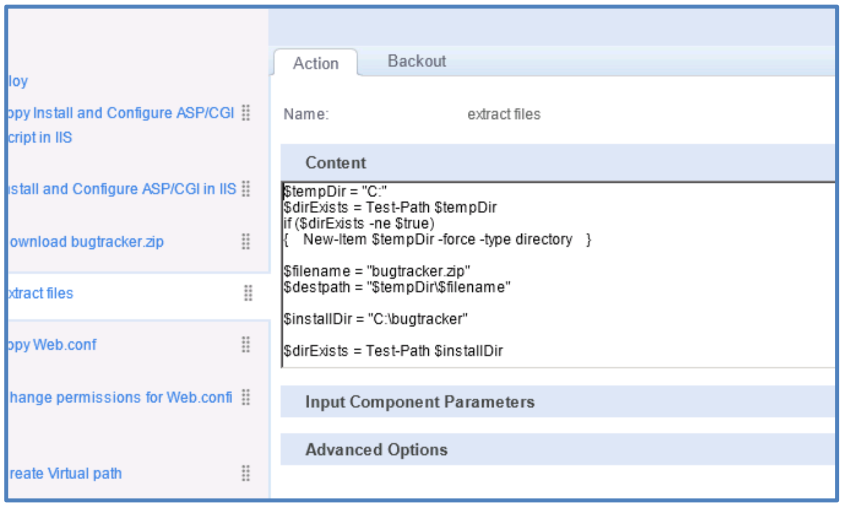
Executed Script Operations

One of the more powerful deployment operations in HP CDA is the Executed Script operation. This allows you to run Shell/Powershell commands directly from HP CDA, or indeed invoke any existing scripts that have been created by any variety of interpreters, such as PERL or Ruby. The approach to use depends on a variety of factors. Consider the following:

- Calling an existing script allows you to maintain one single script in a central location, which may also be in a source/revision control repository such as SVN. In this way, you can ensure that all applications that need this script will be using the same one.
- Executing commands directly from HP CDA as part of your executed script operation allows you to see exactly what is being performed as part of the deployment, without having to open an external script file. You can also create multiple executed script operations in HP CDA, as opposed to invoking the execution of an external script file, which can provide you with more granularity in the event of a failed deployment.

A Helpful Tip mentioned earlier in this document referred to HP CDA sample application models which are available for you to download directly from HP Live Network. The Bug Tracker sample application provides some good examples for building intelligence into your executed scripts, such as making use of conditional statements in executed script operations, calling batch files, etc.:

Figure 6. Conditional Statement in a Bug Tracker Script



loy
opy Install and Configure ASP/CGI
cript in IIS
stall and Configure ASP/CGI in IIS
ownload bugtracker.zip
xtract files
opy Web.conf
hange permissions for Web.confi
reate Virtual path
Action
Backout
Content
Input Component Parameters
Advanced Options

Name: extract files

```
$tempDir = "C:"
$dirExists = Test-Path $tempDir
if ($dirExists -ne $true)
{   New-Item $tempDir -force -type directory   }

$filename = "bugtracker.zip"
$destpath = "$tempDir\$filename"

$installDir = "C:\bugtracker"

$dirExists = Test-Path $installDir
```

When creating executed scripts, you should be aware of the benefits and potential drawbacks of using "Exit 0" statements. "Exit 0" will cause HP CDA to interpret the operation being performed as successful, even if it actually failed. This can be useful in cases such as a Make Directory operation, where if the directory already exists, you don't want your deployment to fail even though the Make Directory operation could not be successfully completed. Similarly, if you instruct HP CDA to start a service, you don't want the deployment to fail if the service is already running.

However, there are downsides to using "Exit 0". For example, HP CDA will also report that the operation succeeded whether it actually did or not, potentially providing a false positive. As an example, the following executed script failed to execute successfully, but HP CDA reported it as passed since an "exit 0" was used (which complicated the troubleshooting process):

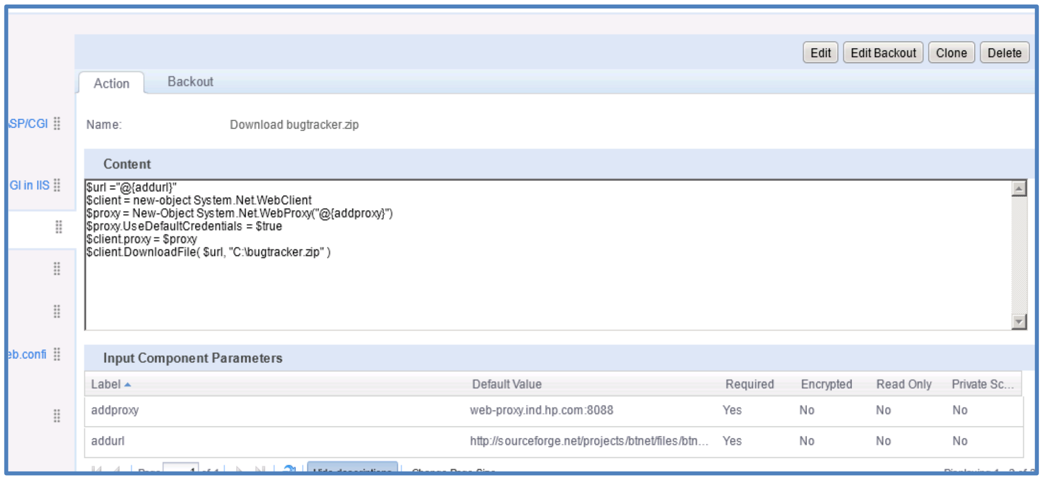
```
#!/bin/sh
apt-get update
apt-get install -y php5
apt-get install -y php5-mysql
exit 0
```

Use HP CDA's "Ignore Failure" option (under Advanced Options) instead of "exit 0". This will allow HP CDA to report the correct and accurate result of each operation. It will, however, ignore any failure and move onto the next step to allow the deployment process to continue onto completion. In this way, you will be able to correctly detect any offending operations, but not have them block you from performing subsequent deployment operations.

Parameterization

Take advantage of HP CDA's parameterization capabilities, so that you can build flexibility and portability into your application models. For example, below we have created input component parameters for two elements of an executed script (proxy and URL). By specifying these input component parameters, their specific values can be provided dynamically by the user at deploy-time. Alternatively, pre-set default values can also be used. You can also use input component parameters in this same way for Placed File operations where you may have a script file, configuration file, or other file that makes use of parameterized content.

Figure 7. Using Input Component Parameters



Name: Download bugtracker.zip

Content

```
$url ="@{addurl}"
$client = new-object System.Net.WebClient
$proxy = New-Object System.Net.WebProxy("@{addproxy}")
$proxy.UseDefaultCredentials = $true
$client.proxy = $proxy
$client.DownloadFile( $url, "C:\bugtracker.zip" )
```

Input Component Parameters

Label	Default Value	Required	Encrypted	Read Only	Private Sc...
addproxy	web-proxy.ind.hp.com:8088	Yes	No	No	No
addurl	http://sourceforge.net/projects/btnet/files/btn...	Yes	No	No	No

Parameterization is also a good approach to enable your applications to cater to multiple users by parameterizing the username and password for applications and middleware that you deploy, or need to interact with.

HP CDA lets you define parameters at multiple levels of your application. For example, you can define parameters at a high level (at the Application version level) for "global" parameters that you may want to reference from any portion and version of your application, and define parameters at a lower level (at the workflow level) for "local" parameters that only one operation may require.

It may also be interesting to explore opportunities to use multiple parameters in your application models such that all specific information related to your application is parameterized. In this way, your application model will be more generic and become a sort of template which can then accommodate the deployment of multiple, similar applications. For example, if you have 10 Apache-based applications which are similar, you may be able to design a single application model which can be used to deploy any of the 10 applications, simply by providing the required parameter values which are specific to each application you want to deploy. This will help keep your application maintenance costs down as well.

Multiple Models

HP CDA supports creating multiple models for a given application. When would you want or need to have more than one model for an application? One instance would be if your application supports multiple databases. For example, a common, cost-saving practice is to use an open-source database management system in the development and potentially QA environments, as there is no license fee incurred. Then, use a RDBMS like Oracle or MySQL in the Staging and Production environments. In such a case, you will most likely need to create two models, one for MySQL and the other for Oracle.

Create your application model for one of the integrations (Oracle, for example) and then use HP CDA's "Clone" feature to create a clone of that application model. Then, simply modify the cloned model as required (say, for MySQL).

Another instance where you may need multiple application models is if your application supports more than one OS version. In this case, your executed scripts would be different, for example, Shell commands for Linux deployments, and Powershell for Windows deployments.

In theory you may be able to abstract some of the OS related differences via parameterization, but in most cases it will be preferable to have separate models for each OS.

There are instances where you do NOT need to create multiple models for your application, and this is one of the benefits of HP CDA's architecture:

- *Deploying your application to a variety of target servers*: Over the application lifecycle, you will most likely need to deploy your application into a variety of environments – for example, a single Tier Development environment, a 2-Tier QA environment, and a multi-Tier Staging or Production environment (where you may have also configured HP CDA for load balancing purposes). You can do this using one, single model, and simply create any number of deployment topologies required, which will associate your single application model with the various target platforms.
- *Using different deployers*: You may choose to use one deployer (SSH, for example) in a Development environment, but then switch to a more robust, fuller-featured deployer like HP Server Automation or the open source Chef tool, in the Staging and Production environments. Here again, this can be accomplished without creating additional application models.

The above approaches help keep the maintenance investment low, as you minimize the number of application models that you will need to maintain and update over time.

Middleware

Many applications require some middleware - for example, Apache, Tomcat, .NET, etc. - to be installed on the target server in order to run. HP CDA provides you with the flexibility to define your application, along with any required middleware, in the way that best suits your needs. The recommended approach would be to make use of HP CDA's Platform Software feature to define the middleware required by your applications. Platform software can be defined independently of the application, allowing middleware models to be used by any number of applications. These leverage and re-use capabilities will greatly minimize the time required to create new application models over time.

In most cases, it will be advantageous to model middleware at the Platform level, for the following reasons:

- This allows middleware to be modeled in a modular fashion, so it subsequently can be mixed and matched with any number of apps, which in turn provides you with more flexibility and allows you to build up a software library over time;
- Provides for more efficient maintenance. You can maintain once in a single, central model, rather than in all of the individual application models that may be using the particular middleware;
- Shortens your application deployment time. You won't need to deploy platform software each time you want to deploy or redeploy your application, speeding up deployments.

Alternatively, you may decide to define your middleware model as part of your application model. This could be applicable in some cases – for example, if only one application will be using a given middleware package - or for applications, custom applications or homegrown apps which may not require mainstream middleware. And while this is not, strictly speaking, the recommended approach, HP CDA provides you with the flexibility to design your application model in this way, if you so choose.

Make use of HP CDA's workflow invocation capabilities to allow you to design your application model so that it can invoke middleware installation/configuration operations dynamically, at any point during your application deployment.

You may also want to consider including ("baking in") any required middleware into a template that can be used when provisioning servers, instead of using HP CDA to deploy your middleware. Here again, there are pros and cons associated with using pre-installed ("baked in") middleware versus deploying middleware at provision time. With pre-installed software, your infrastructure provisioning operations will be faster, as you won't have to deploy middleware at provision time. You can also perform any configuration operations up front, so that when HP CDA provisions your platforms they will be pre-configured and ready for application deployments. The downside of using the pre-installed software approach is that you will not build up a library of middleware within HP CDA, potentially impacting your leverage and re-use abilities. You will also potentially need to manage and maintain more templates, as you will not have basic OS images, but rather several images with various combinations of middleware pre-installed. The pre-installed approach may have eventual license (and hence cost) implications as well.

Look into and exploit any opportunities you may have to leverage existing content for your middleware deployment requirements. For example, there may be DMA flows (HP Database and Middleware Administration), OO flows (HP Operations Orchestration), SA scripts, packages, or policies (HP Server Administration) or even Chef recipes that you can invoke for middleware deployment. You can invoke any of this content dynamically, at any point during your application deployment.

Use HP CDA's Platform Software capabilities to do more than just deploy middleware. For example, you can designate that HP CDA perform any additional configuration operations or test setup tasks at the very end of deploying middleware. You could also deploy any test files or test harness files in preparation for ensuing tests that will be conducted after the actual application and middleware has been deployed.

Application Deployments and Troubleshooting

To achieve maximum control over how you deploy your applications and what you deploy, make use of HP CDA's Options capabilities. These capabilities allow you to choose selected portions of your application model to deploy. This gives you the ability to deploy only certain portions of the overall application, depending on your needs. For example, if you are a developer or tester and need to deploy an application to do a basic smoke test, you can design your application model to deploy only the core application. Then later, when you want to perform broader testing, you can instruct HP CDA to deploy additional portions (for example, a help sub-system, language packs, etc.). Note that you can achieve this also by directly mapping your application layers in the Deployment Topologies screen using HP CDA's intuitive drag-and-drop feature.

Another very useful feature which gives you very granular control over your application deployment operations is HP CDA's operation Enable/Disable capabilities. Using an intuitive toggle, you can select which specific operations will you to execute (enable) during your deployment, and which you temporarily want to skip (disable). This is very helpful when building your application models for the first time, and when troubleshooting and debugging issues. In this way, if you are debugging a deployment error and your application model is comprised of 20 individual operations, you can disable all but the offending operation so that you can focus quickly and directly at the operation which needs to be addressed.

Helpful Tips

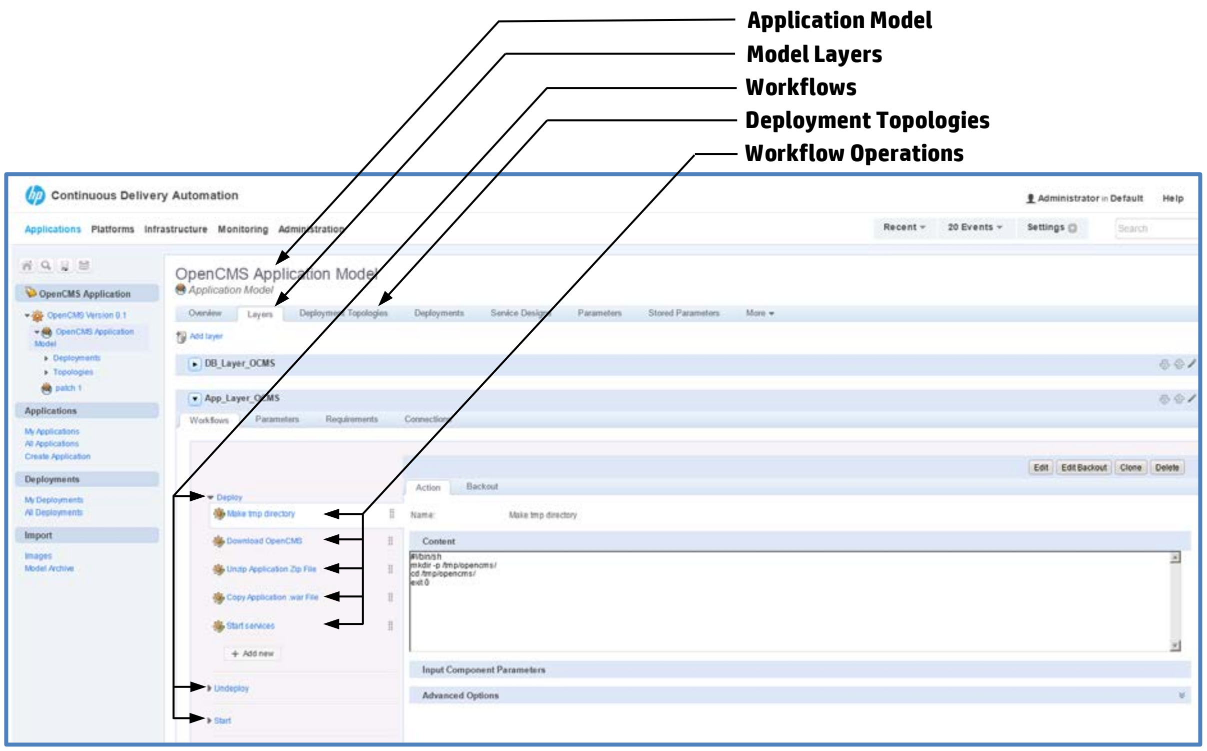
Use HP CDA's job timeouts (for deployments, provision operations, etc.) so that you can instruct HP CDA to terminate any rogue processes that may be causing deployment failures.

As mentioned above, be sure to provide Undeploy and Backout steps as part of your application model. This is important for troubleshooting after a failed operation, as you will want to retry your deployment after resolving the cause of the issue, and you will want to do this with your target server in a known good state.

Helpful Tips

If an executed script operation fails as part of your deployment, try executing the failed command directly in Powershell (or Shell if Linux). This will help you narrow down the reason for the failure.

Terminology used in this White Paper



```
#!/bin/sh
mkdir -p /tmp/opencms/
cd /tmp/opencms/
exit 0
```

For more information

HP software product manuals and documentation for HP CDA can be found at http://h20230.www2.hp.com/selfsolve/manuals. You will need an HP Passport to sign in and gain access.

Note
General-access documentation requires that you register for an HP Passport and sign in. In some cases, access to the documentation is restricted and requires that you have an active HP support agreement ID (SAID) and an HP Passport sign-in.

Table 1. Document Revision History

Date or Version	Changes
November 2013	Initial release of document.

Learn more at
hp.com/go/cda

Sign up for updates
hp.com/go/getupdated

November 2013