Configuring Downloadable Objects for Cisco Videoscape Voyager Vantage

Please Read

Important

Please read this entire guide. If this guide provides installation or operation instructions, give particular attention to all safety statements included in this guide.

Notices

Trademark Acknowledgments

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at **www.cisco.com/go/trademarks**.

Third party trademarks mentioned are the property of their respective owners.

The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1009R)

Publication Disclaimer

Cisco Systems, Inc. assumes no responsibility for errors or omissions that may appear in this publication. We reserve the right to change this publication at any time without notice. This document is not to be construed as conferring by implication, estoppel, or otherwise any license or right under any copyright or patent, whether or not the use of any information in this document employs an invention claimed in any existing or later issued patent.

Copyright

Contents

About This Guide

Purpose

This guide provides information to service providers on how to create and deploy applications, themes, logos, and language packs using the Object Download feature of the Cisco Videoscape™ Voyager Vantage (Vantage). Systems that use the Object Download feature of Videoscape Voyager Vantage must currently support version 3.0 or later of Vantage code.

Audience

This guide is written for service providers, headend operators, and field service engineers who are responsible for using the Object Download feature of Vantage.

Read the Entire Guide

Please review this entire guide before using it. If you are uncomfortable with any of the information, contact Cisco® Services at 1-866-787-3866 for assistance.

Required Skills and Expertise

System operators or engineers who install or upgrade Digital Network Control System (DNCS) software need the following skills:

- Advanced knowledge of UNIX
  - Experience with the UNIX vi editor. Several times throughout this installation process system files are edited using the UNIX vi editor. The UNIX vi editor is not intuitive. The instructions provided in this guide are no substitute for an advanced working knowledge of vi.
- Extensive DNCS system expertise
- The ability to add and remove user accounts

Document Version

This is the first formal release of this document.

1

Overview of the Object Download Feature

Introduction

The Object Download feature of Vantage provides a means for applications, themes, logos, and language packs (objects) to be downloaded to the set-top box (STB). This chapter provides a definition of some terms commonly associated with downloadable objects, and shows a few examples of how the Object Download feature is implemented for subscribers.

Subsequent chapters in this book cover the following items:

- The contents of the configuration file (config.xml) that define the parameters of the Object Download feature, and how the objects integrate themselves into the user interface
- How system operators deploy the application, theme, logo, and language pack objects
- How to authorize and provision STBs for downloadable objects

In This Chapter

Definitions

The following terms pertain to the Object Download feature:

- **Application** — downloadable widget-style tool, such as a weather widget, stock ticker, or an email reader
- **Theme** — customized look-and-feel to brand the user interface with colors and images
- **Logo** — insignia of the service provider
- **Language Pack** — package to download a new language set into the user interface
- **Walled Garden** — a network-hosted portal application, such as a News Portal, Games Portal, or Self-Service Provisioning Portal

Push or Pull Downloadable Objects

Applications, themes, logos, and language packs are downloaded to STBs by the following two methods:

- They are downloaded *automatically* (**pushed**) to STBs by the service provider through means of the Broadcast File System (BFS) server of the DNCS.
- Subscribers download these items at will (**pulled**), by means of HTTP, to their STBs through an Application Store Walled Garden.

  Note: This method requires that the service provider implement and operate a Web Portal Application Store through a Walled Garden configured on the DNCS.

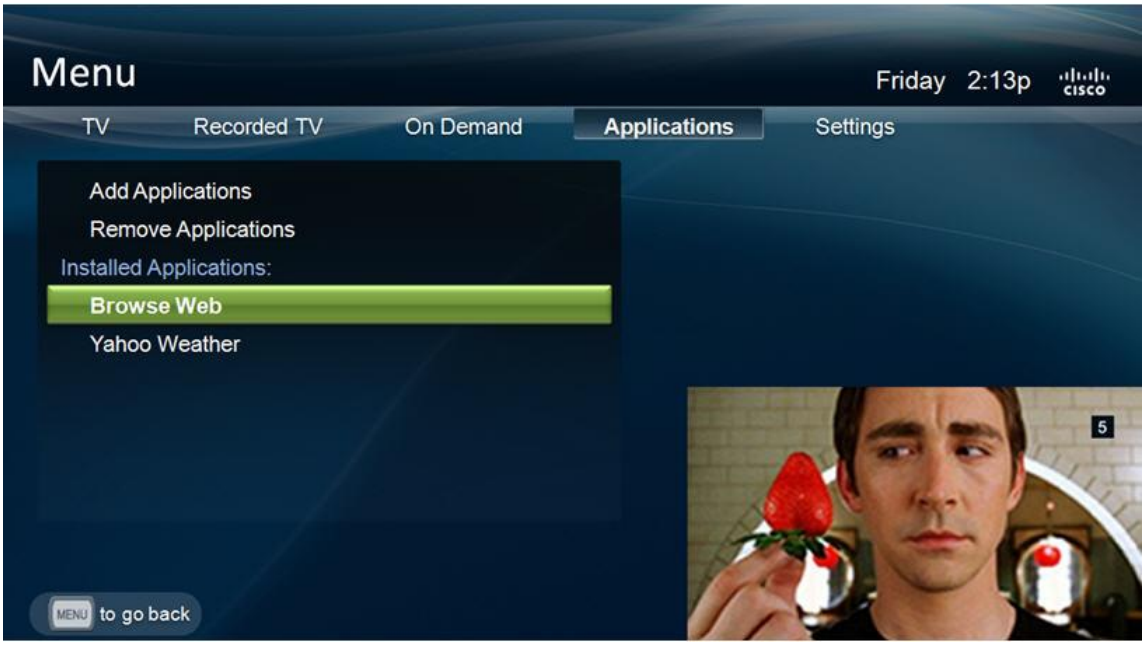
Both of these methods require configuration on the DNCS.

The Subscriber Interface

This section presents examples of how subscribers access the various applications and settings through their STB.

Main Menu

- The **Applications** menu item displays a list of applications and Walled Gardens available to the user.
- The **Settings** --> **Appearance** menu item displays a list of themes available to the user.
- The **Settings** --> **System** menu item displays a list of screen languages available to the user.



Notes:

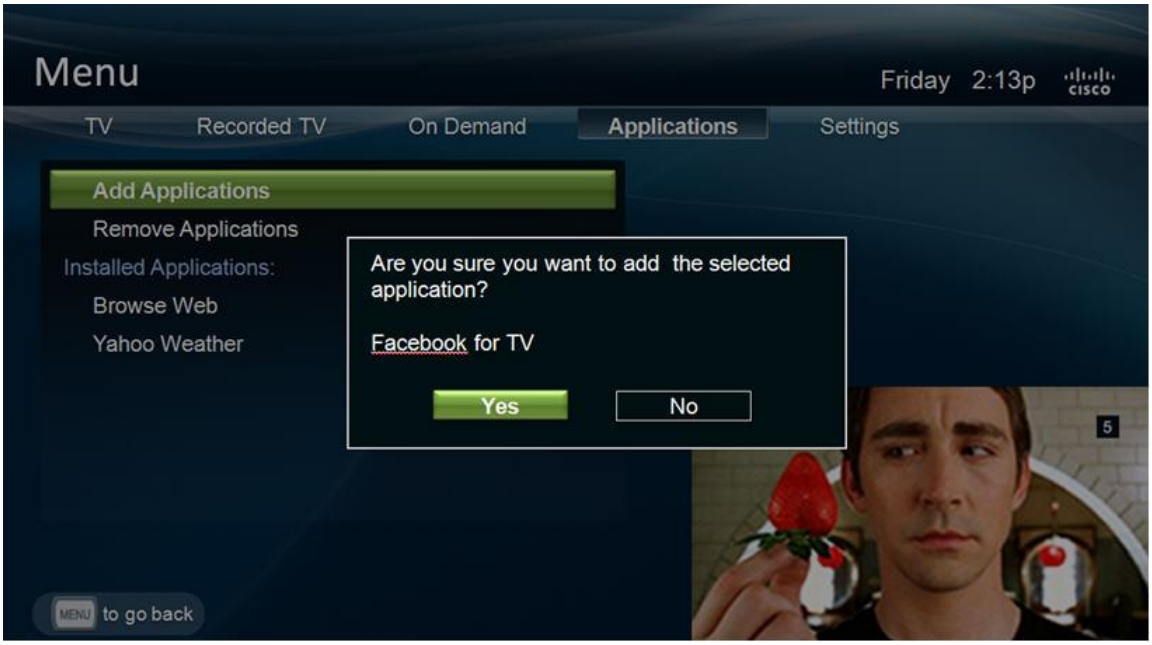
- The **Installed Applications** menu item represents the current list of installed (push or pull) applications and provisioned Walled Gardens that are configured to appear in the Applications menu (through the config.xml file).
- Selection of **Add Applications** leads to the Application Store provisioned by the system operator (if the Walled Garden _WGAP is provisioned).
- **Remove Applications** allows the subscriber to remove previously pull-downloaded objects.

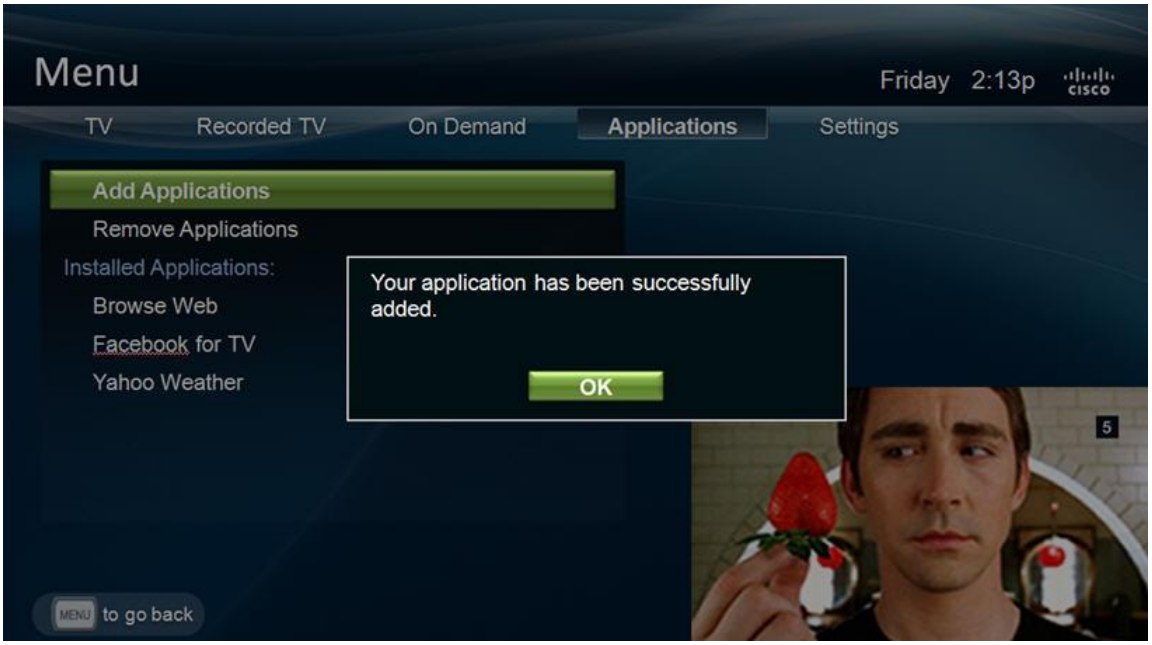
Available Applications (Through the Application Store)

The following image is an example of a Walled Garden Application Store. In this environment, users can pull-download applications. Support for pull-downloaded applications is dependent upon operator configurations.

Add Applications

The following two menu images represent examples of the use of the **Add Applications** menu option as if the subscriber had accessed an Application Store provisioned by the system operator and selected to download the *Facebook for TV* application.





Remove Applications

The subscriber chooses an application to remove from the STB.

Note: Only applications that were pull-downloaded by the subscriber are available to be removed from the **Remove Applications** list. Applications that are push-downloaded by the operator are not available for removal by the subscriber.

2

Creating a Downloadable Object

Introduction

This chapter helps software developers accomplish the following tasks needed to create a downloadable object:

- Develop the required configuration file (config.xml) with a text editor
- Develop the package(s) using the configuration file
- Bundle the package(s)

In This Chapter

The config.xml Configuration File

The config.xml file defines all the metadata for a downloadable object, and must be provided with an object in order for the object to be valid. The file is always called config.xml and always resides in the root of the object file tree. It has many properties, most of which are optional. You can create the config.xml using the text editor of your choice.

This section provides an overview of the config.xml file, as well as a description of the fields required for the file.

Example of the config.xml File

An example of the config.xml follows. Elements marked with an asterisk (*) can be repeated in the file.

```
<?xml version="1.0" encoding="utf-8" ?>
<package id="" type="suite-app|suite-theme|suite-
language|suite-logo">
      <name lang=""></name>*
      <description lang=""></description>*
      <uri></uri>
      <author></author>
      <version></version>
      <orgId></orgId>
      <galioVersion></galioVersion>*
      <property
            name="launch|show|destroy|showLoadingBanner

      |resolution|forceUpgrade|lowMemProtect|langCode2|*">
      </property>*
      <menu show="yes|no"></menu>
      <group></group>
```

```
      <key></key>*
      <iconUri width="" height="" lang=""
variant=""></iconUri>*
      <channel displayInGuide="yes|no"></channel>
      <licence></licence>
      <memoryRequired malloc="" flex=""></memoryRequired>
      <feature required="yes|no">dvr|*</feature>
</package>
```

Objects must have, at a minimum, the following attributes and elements or else the object is considered to be invalid and is ignored:

- Attributes
  - id
  - type
- Elements
  - version
  - orgId

Attributes

<package id>

The package id of the object must be unique within the organization. It will be combined with the orgId and the version string to create a globally unique object reference.

Note: The package id must be 4 characters or less, such as "ynws".

<type>

The type attribute must be one of: "suite-app", "suite-theme", "suite-language", or "suite-logo", depending on the object type.

<lang>

The lang attribute provides the language for the text element in the form used by the user interface, such as eng_us, eng_gb, spa, hin, ara, hrv, ita, por, and/or por_br. This is provided in the name, description, and iconUri fields, as shown in ***Example of the config.xml File*** (on page 10).

Elements

<name>

This is the localized name of the object as displayed to the user in user interface menus and when referenced in the UI. The lang attribute is used with the <name> element. Multiple <name> elements can be defined, one per language.

<description>

The localized long description of the object is displayed to the user in a free text format. The lang attribute is used with the <description> element. Multiple <description> elements can be defined, one per language. The description element is not displayed on the user interface.

<uri>

The uri is the HTML file used to launch the application as a relative URI. This should nearly always be <id>.html (e.g. twit.html) and defaults to this if not given.

<author>

This element provides author/contact details for the object. It is in free text format, and can be a name, email address, etc. The author element is not displayed on the user interface.

<version>

This is a string that is used to display the object version to the user but will also be compared against other version strings to determine whether a new object is an upgrade or not. For this comparison it is assumed that the string consists of versions and sub versions separated by dots. The string is compared numerically rather than textually. For example, version 1.2 is older than 1.10.

<orgId>

This is the id of the organization that created this object. Object ids are assumed to be unique within an organization. Therefore, an id and orgId together provide a globally unique object name. This should be a domain style name, such as com.antplc or com.cisco.itr.

<galioVersion>

Galio is the STB web browser used in conjunction with the Object Download feature of Vantage. The galioVersion element is the required version of Galio for a specific package. For example, a galioVersion of 3.0.2 indicates that this package requires any version of Galio 3.0 maintenance release 2 or later, or any version of Galio 3.1 or later. This element can be included multiple times to allow for custom Galio releases on different branches that must be supported. Applications specifying a later version of Galio than the version currently in use are not visible to the UI and are treated as if they do not exist.

Note: The current version of Galio is 3.1.5.

<property>

The property element adds arbitrary name/value pairs that are interpreted only by the UI.

The following table lists the property elements currently in use:

Name	Possible Values	Default	Description
launch	yes/no	yes	Should the application be automatically launched on startup?
show	yes/no	no	Should the application be automatically shown when launched?
destroy	yes/no	no	Should the application be killed when hidden?
showLoadingBanner	yes/no	no	Should "Loading…" banner be shown when opening application?
resolution		None	Describes the resolution that a theme provides. (Usually 1280x720 or 640x480)
lowMemProtect	yes/no	no	Should the application be protected from being killed in free memory when Vantage is running low on memory?
forceUpgrade	yes/no	no	If set to yes then the user is not given the option to defer an upgrade to a more convenient time. It must occur immediately.

Name	Possible Values	Default	Description
langCode2	en, es, ko, ...	None	Used on language packs only. The 2-letter version of the ISO-639 language code. If the language uses a "right to left" script, then it should include the suffix :rtl, for example: ar:rtl

<menu>

The menu element controls whether the application is displayed on the Applications menu and allows for the position on the menu to be fixed. If the element is not provided, then the application is shown on the Applications menu in the default location.

Lower positions mean higher locations on the menu. Values less than 100 are reserved for built-in applications and upgrades to those applications. If a broadcast or downloaded application uses a position less than 100, then it is treated as if it did not provide a position.

The order is undefined if two applications use the same position.

<group>

This is the group number for application suite objects. This value should always be 200 in order to avoid multiple applications from being visible at the same time.

<key>

Any number of key definitions can be provided, including none. These are keys that, if pressed, will launch the application (not valid for theme and language).

The element should contain a string as defined in keyboard.js (without the TVLib.Keyboard prefix), such as KEY_M or KEY_DEVICE_YELLOW.

When a key is pressed, a matching built-in application (or upgraded built-in application) is searched. If no match is found, then the broadcast and downloaded applications are searched. This ensures that downloaded applications cannot hijack the important keys like GUIDE, DVR, etc.

<iconUri>

Multiple icon files can be provided for different sizes, languages, and variants. The best match will be used by the application suite. These URIs are usually relative to the root of the object, but absolute URIs can be given instead.

The variant string is either "" (the un-themed icon), "theme" (the un-highlighted icon for theme) or "theme_f" (the highlighted icon for theme).

The iconUri element is not implemented in the user interface at the present time.

<channel>

The channel element is optional. If provided, a virtual channel is created for this object, such as 5000. Only applications and web pages can be assigned virtual channels. Note that channels specified in an application's config.xml file can only be used if they are not defined by the Channel List Manager (whether initially or due to a channel list update).

<license>

The license element provides license details for the object in a free text format. The license element is not displayed in the user interface.

<memoryRequired>

The memory element represents the expected maximum amount of memory this application requires to run (including all unshared images). It is specified in bytes in the malloc and flex memory areas. If present, this element is used to ensure that there is enough memory for the application before running it. This makes loading quicker and more reliable.

<feature>

By using the feature element, an object declares that it uses this feature. If the required attribute is set to "yes" then the object cannot run unless that feature is present. If the feature is not present, then the object is ignored and is not presented in the Vantage UI. The set of features defined at present is as follows:

Name	Description	Use
dvr	The hardware has local recording capability	If DVR functionality does not exist, do not show this object.

Developing Objects

Application Packages

The following example demonstrates the layout of an application object. The config.xml file contains the location of the start file which is normally <app>.html. The messages (for language localization) and theme definitions (for application skins) can be provided in the application object or in language/theme objects.

The type attribute of the config.xml file must have the value "suite-app".

```
config.xml
<app>.html
js
      ...
      messages
            ...
css
      ...
      themes
            <theme>*
                  ...
```

Theme Packages

A theme object contains all the necessary library files for the theme. It can optionally contain application-specific theme files as well.

The type attribute of the config.xml file must have the value "suite-theme".

The following example demonstrates the layout of a theme object.

```
config.xml
library
      css
            themes
                  <theme>
                  ...
<app>*
      css
            themes
                  <theme>
                  ...
```

Logo Packages

A logo object typically contains operator logos that are displayed on applications for branding. The logos may be customized for particular themes or may be generic and usable on any theme. Having them in a separate object allows simple customization of one theme for multiple operators without having to rebuild the theme object (although that is always available as an option).

The type attribute of the config.xml file must have the value "suite-logo".

The layout of the logo object is not tightly defined and can be changed depending on the themes that will access it. An example that provides logos for two different theme sizes (repeated for each theme) and an un-themed logo is laid out as demonstrated in the following example:

```
config.xml
operator.png
<theme>*
      1280_720
            operator.png
      640_480
            operator.png
```

Language Packages

A language object contains the global dictionary file for the specified language and may also contain translation files for one or more applications. It may contain language-specific CSS files for themes.

The type attribute of the config.xml file must have the value "suite-language".

```
config.xml
library
      js
            messages
                  <lang>.js
      css
            themes
                  <theme>*
                        global_<lang>.css
                        global_<lang>_<www>_<hhh>.css
<app>*
      js
            messages
                  <lang>.js
      css
            themes
                  <theme>*
                        <app>_<lang>.css
                        <app>_<lang>_<www>_<hhh>.css
```

Theme and Language File Loading Order

The following four files are found in the library/css/themes/<theme> directory of the object for the current language. If not found there, then the system searches the object for the current theme next. Links to whichever are found first are used by the application.

- global.css
- global_<www>_<hhh>.css
- global_<lang>.css
- global_<lang>_<www>_<hhh>.css

The following four files are found in the css/themes/<theme> directory of the application object. If not found there, then the system searches in the <app>/css/themes/<theme> directories of the current language object. Finally, the current theme object is searched. Links to whichever are found first are used by the application.

- <app>.css
- <app>_<www>_<hhh>.css
- <app>_<lang>.css
- <app>_<lang>_<www>_<hhh>.css

The global dictionary is loaded from library file /js/messages/<lang>.js in the current language object. The application dictionary is loaded from the js/messages/<lang>.js directory in the application object or, if not found, the <app>/js/messages/<lang>.js file in the current language object.

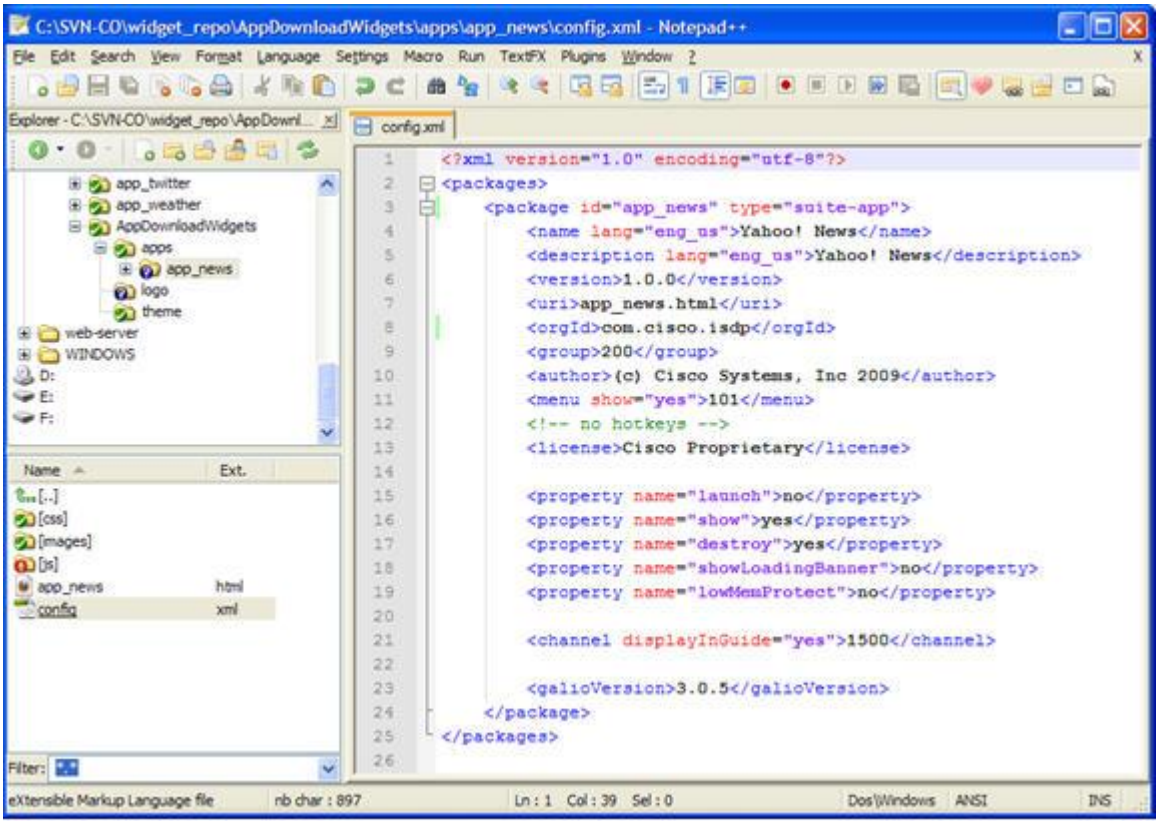
- <app> - current application
- <lang> - current language
- <www> - current screen width in pixels
- <hhh> - current screen height in pixels

Bundling Objects

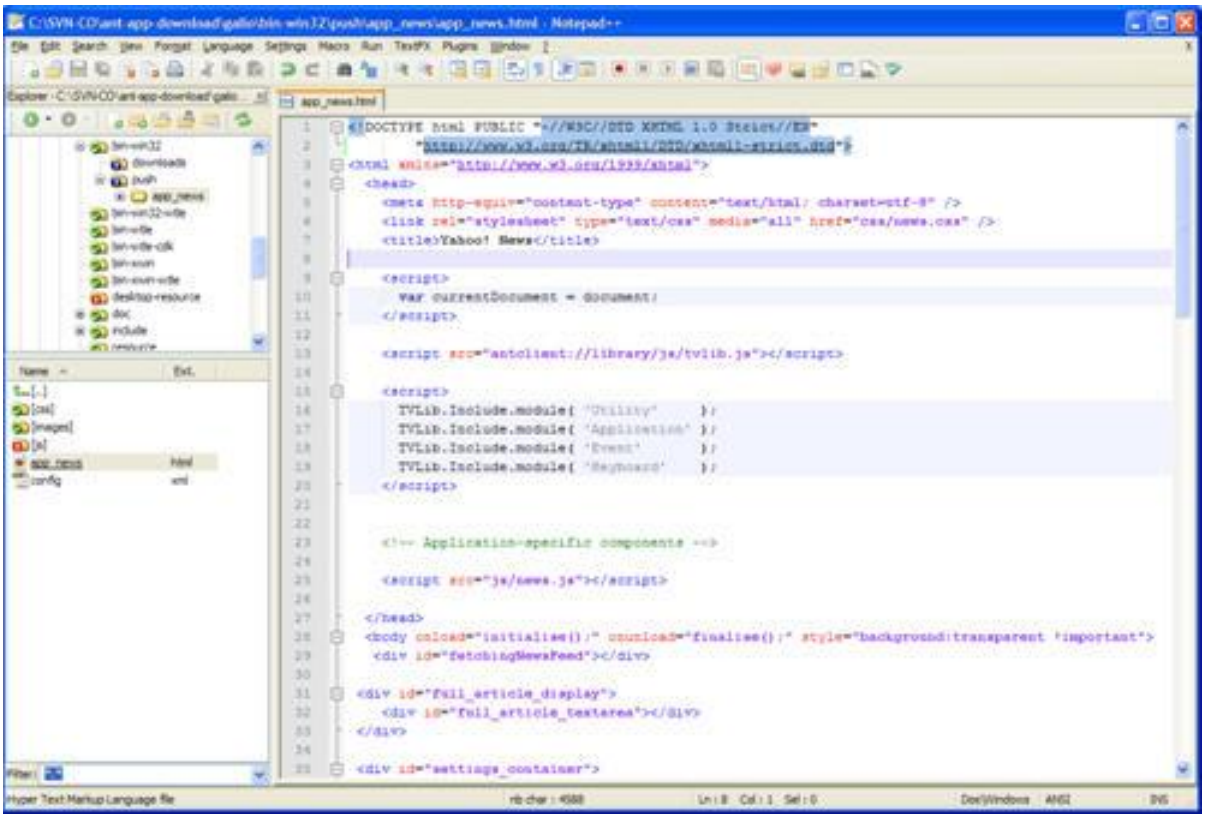
Once the config.xml file and the objects have been developed, the developer can go about the steps of bundling the object into a downloadable object to place on the BFS carousel (for push download) or on an HTTP Walled Garden Applications Store.

Bundling Applications

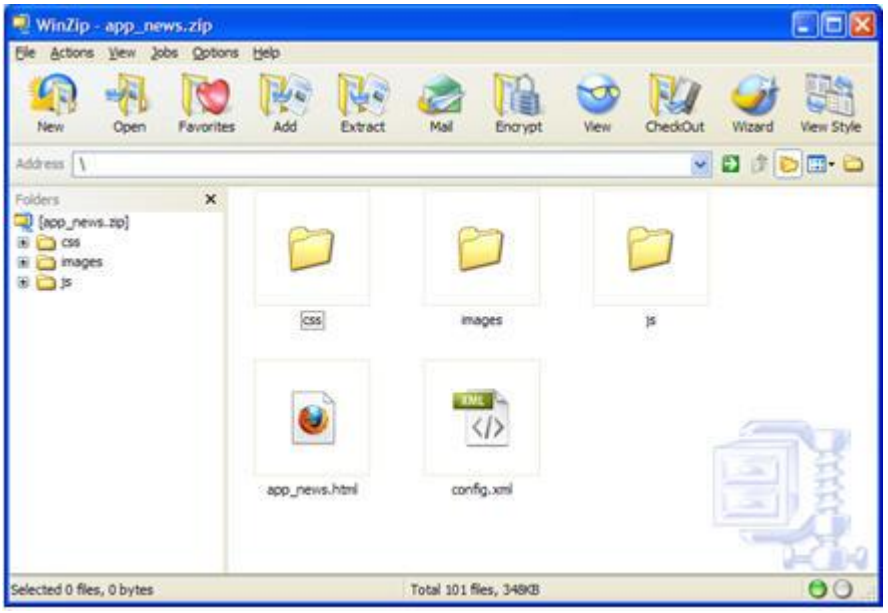
The accompanying screenshot displays the "app_news" application and its corresponding config.xml file. In this application, the app_news.html file is the initial launch file and the config.xml file is included with settings as described earlier in this chapter. This application will appear at location 101 in the Applications menu (just under the built-in applications) and as channel 1500 in the Electronic Program Guide (EPG). It will not launch when the set-top boots. Instead, it will be displayed when launched by the user. It will be destroyed to free up memory when the user exits, and it will not show the loading banner. It is not protected upon Vantage low-memory events; it can be destroyed if memory is needed.



The following screenshot references the common library element using the antclient://library/js/tvlib.js tag, followed by instantiations of some of the TVLib (Vantage) modules. For example, you can see the **Utility**, **Event**, **Application**, and **Keyboard** modules.

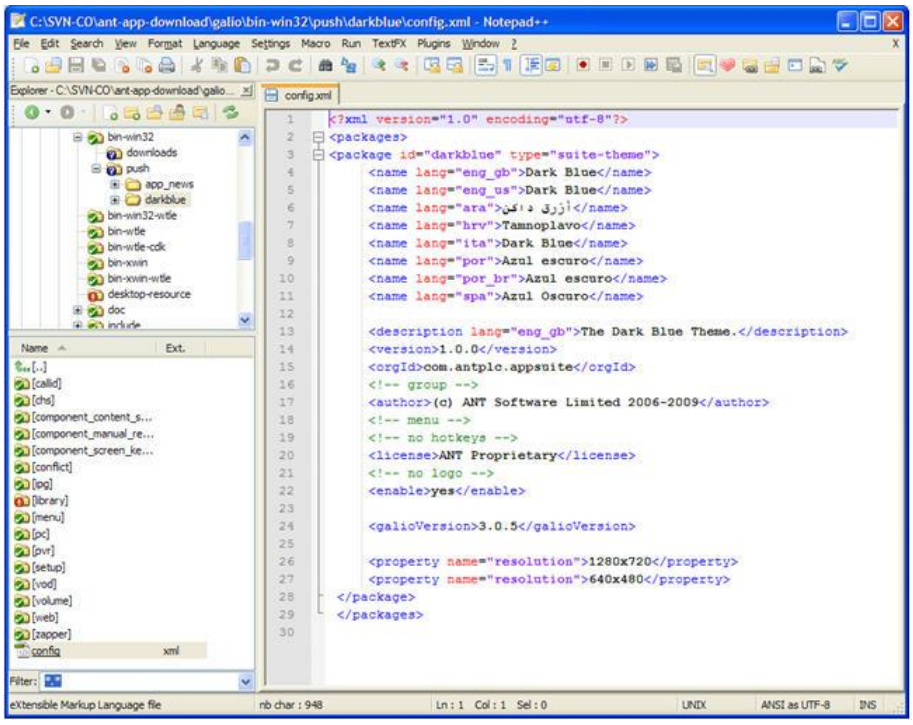


This object can then be compressed (zipped) and added to the BFS (for push) or a walled garden server (for pull) for distribution to subscribers. After compression, the app_news.zip file might look like the following example:

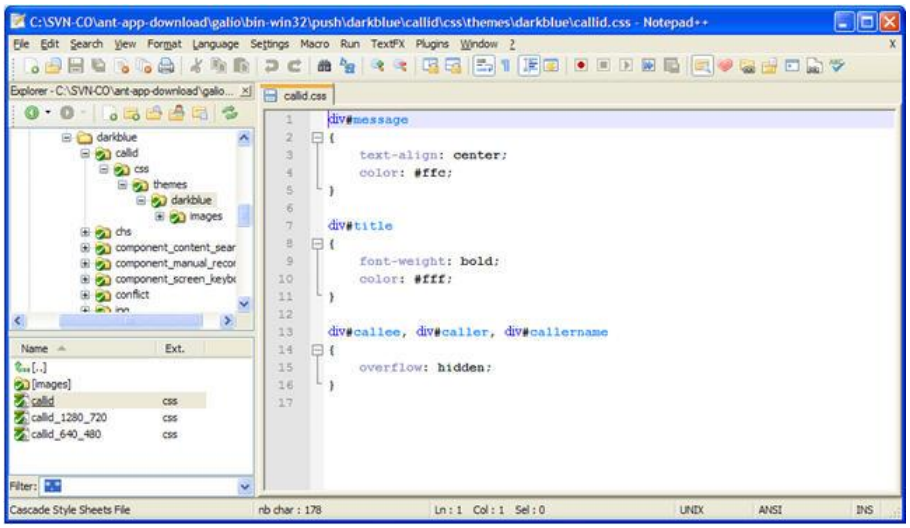


Bundling Themes

Notice in the following example how the **darkblue** theme and its corresponding config.xml file are displayed. In this theme, each application has its own folder. Each folder has a css sub-directory. Each css sub-directory contains its own css and image content files.

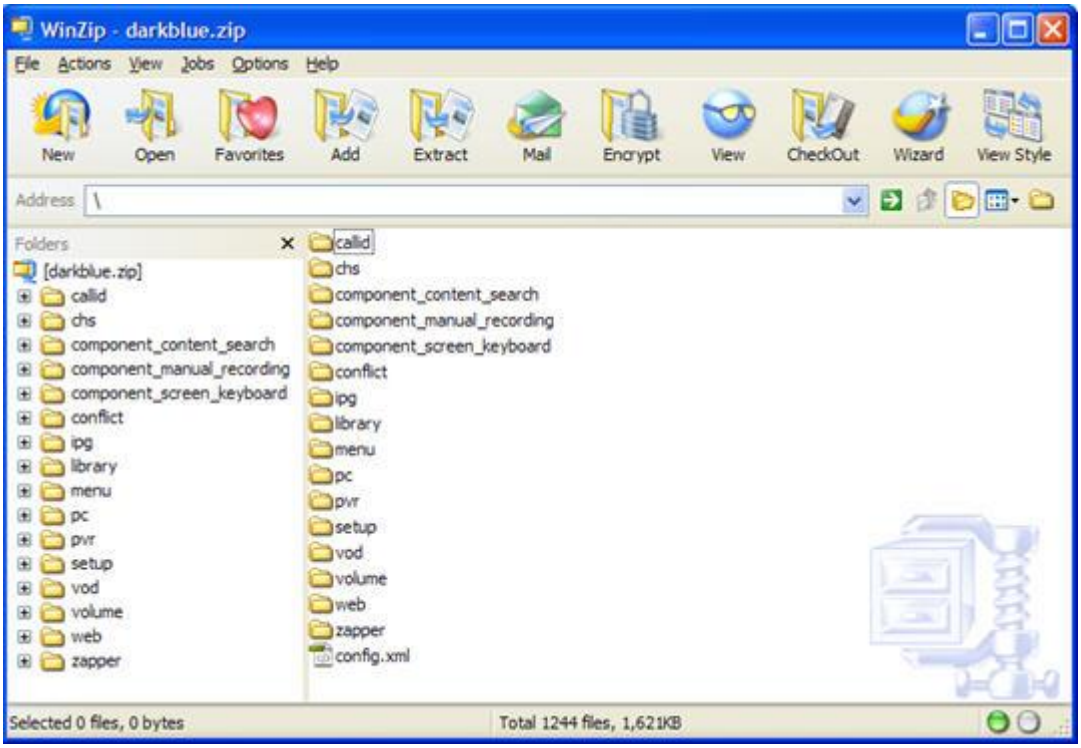


Notice the expanded **callid** application theme defined in the following example:



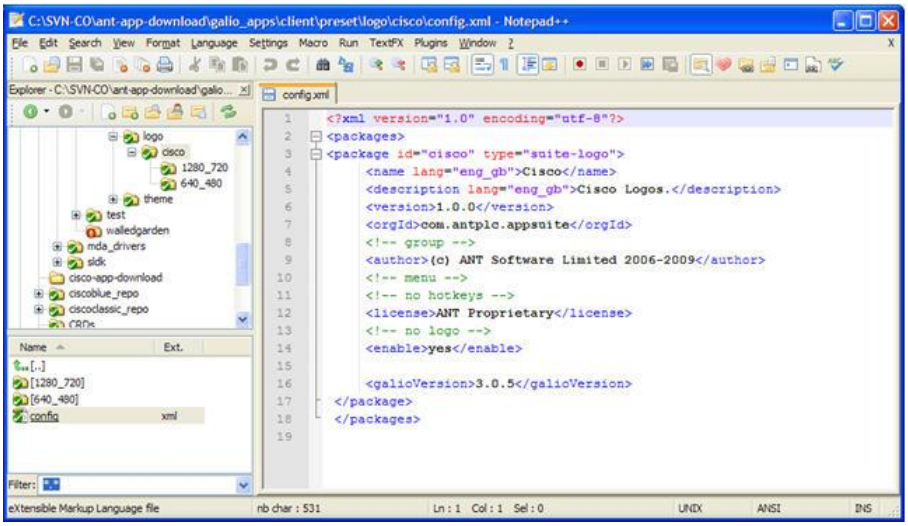
The darkblue theme for the callid application contains the following files: callid_1280_720.css, callid_640_480.css, callid.css. An image folder contains the image assets for the callid application using the darkblue theme.

This object can be compressed and added to the BFS or to the walled garden for distribution to subscribers. The compressed darkblue.zip file might look like the following example:

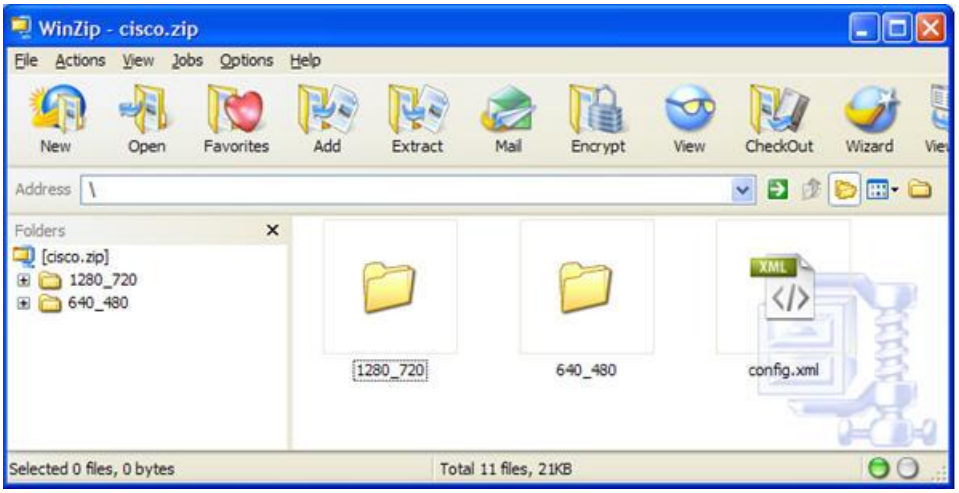


Bundling Logo Packs

The following example shows the Cisco logo and its associated config.xml file. With logo packs, each resolution includes its own folder. Each folder contains the operator.png file of the operator's logo in each directory.

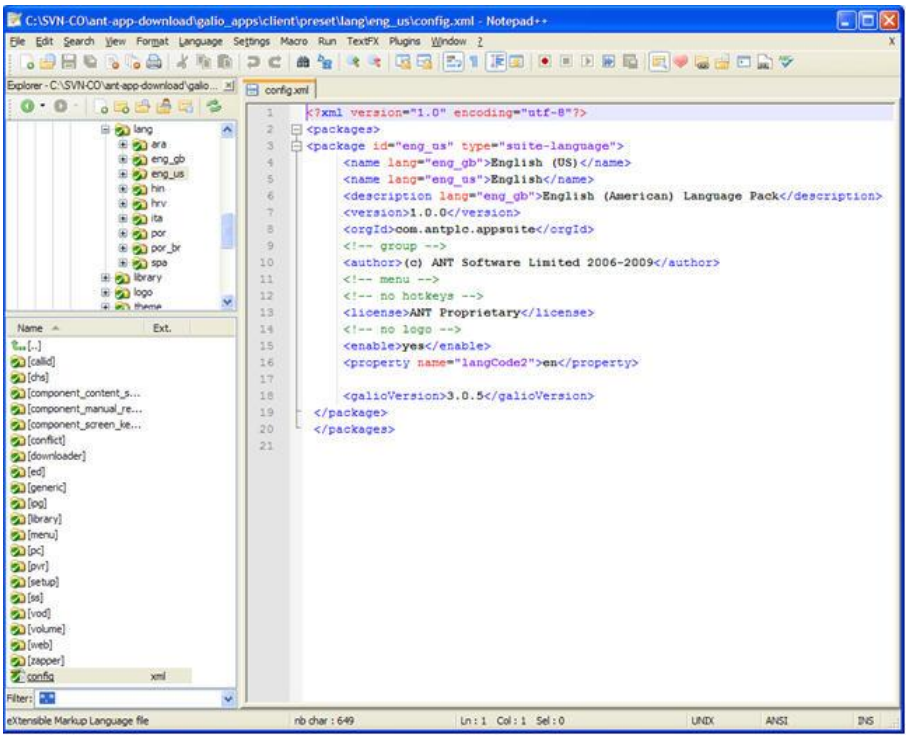


This object can be compressed (zipped) and added to the BFS or walled garden web server for distribution to subscribers. The cisco.zip file might look like the following example:

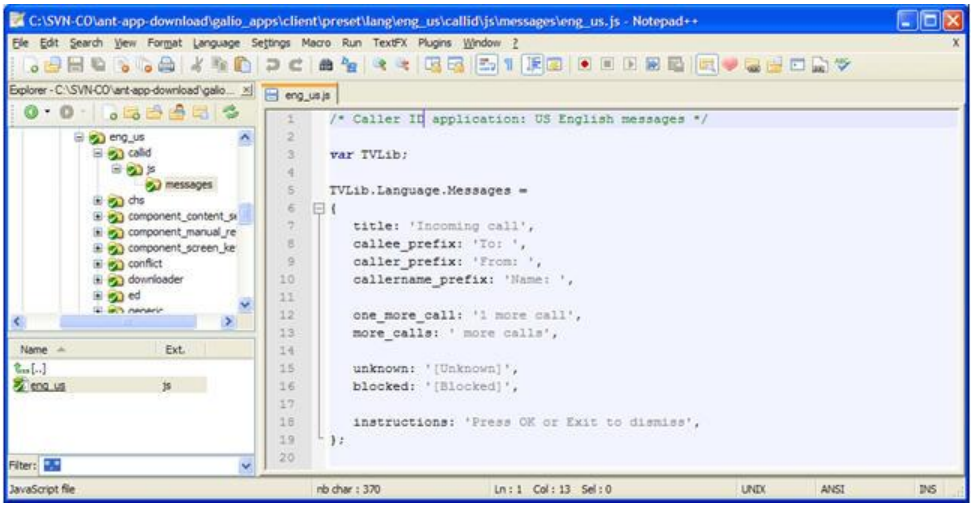


Bundling Language Packs

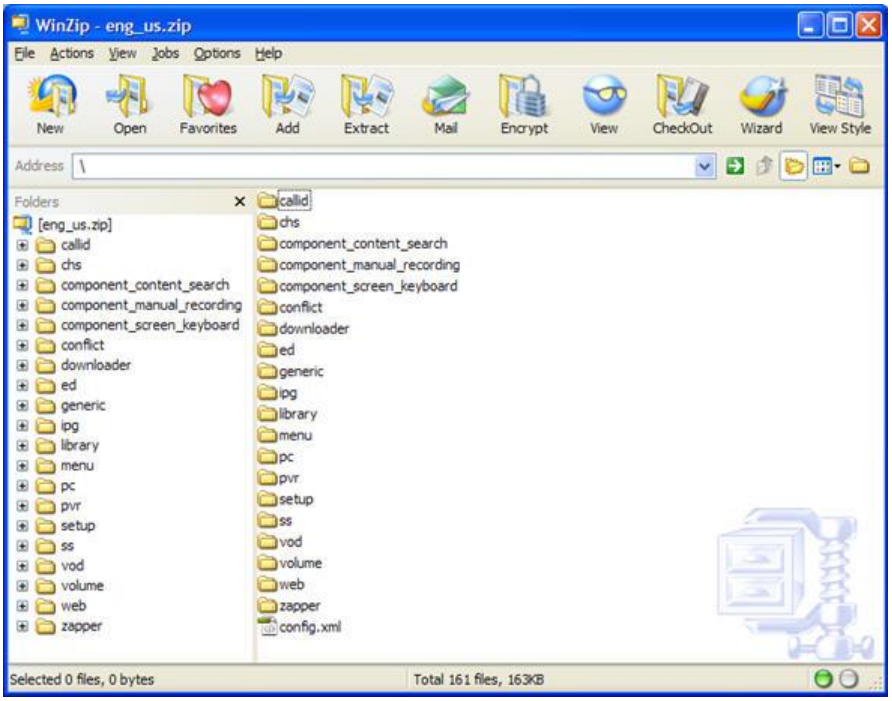
In the following example, notice the American English Language Pack and its associated config.xml file. With language packs, each language is its own object, and, just like with themes, each application has its own sub-directory. In each application sub-directory, there are js (javascript) and message sub-directories that contain the javascript file, which details all of the messages and languages associated with that application.



Note the expanded callid application language defined in the following example.



This object can be compressed (zipped) and added to the BFS or walled garden web server for distribution to subscribers. The eng_us.zip file might look like the following example.



3

Deploying Application, Theme, Logo, and Language Packages

Introduction

After creating and bundling the downloadable objects, you are now ready to deploy them. The material in this chapter guides you through the process.

In This Chapter

Overview of Package Deployment

Upon startup, Vantage checks the following two locations for downloaded objects:

- The BFS directory on the DNCS for broadcast (push) updates. This is a pre-configured location managed entirely by the DNCS.

  Note: The default BFS location for placing downloadable objects is bfs://rtnclient/galio/packages.

- The download cache for objects explicitly fetched by the user. This is a pre-configured, persistent location on a local file system (HDD or Flash).

Vantage then resolves the list and only presents objects to the user interface that are provisioned for use by the STB and that are compatible with the version of Vantage running on the STB.

While Vantage is running, it polls the broadcast directory for any changes in content and then signals the user interface. Where multiple copies of the same object exist across the directories, only the copy with the latest version number appears to Vantage.

The Push Model for Deployment

The push model of downloadable object deployment allows an operator to send objects to the STB without providing a choice to the end-user. This approach may be used to deploy "red button" applications (applications tied to a specific show or channel), to deploy updates to built-in applications or themes, or to deploy other applications distributed by the operator. In order to use the push model for deployment, the operator simply places the compressed (zipped) objects into the BFS directory at bfs://rtnclient/galio/packages.

Example: bfs://rtnclient/galio/packages**/ynws.zip**

The operator must then authorize eligible STBs on the network for the package using the billing system with the SAM service name "_<object id>".

Example: _ynws

Note: Logo objects and upgrades to built-in objects, such as ipg, pvr, vod, or built-in themes, do not require authorization.

Creating the bfs://rtnclient/galio/packages BFS Directory

Complete these steps in order to create the bfs://rtnclient/galio/packages directory.

Notes:

- Steps 1 through 19 need to be completed only once.
- Steps 20 through 23 needs to be completed for each object you add to the BFS.

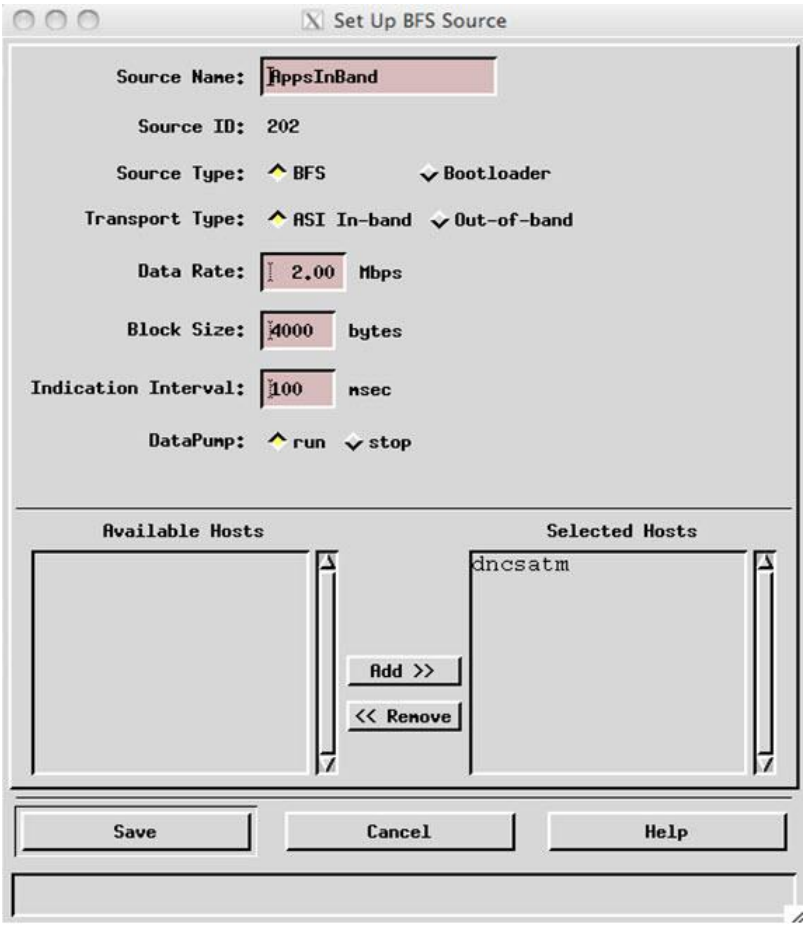
1. From the DNCS Administrative Console, click the **Application Interface Modules** tab.
2. Click **BFS Admin**. Verify that under the **Hosts** tab, the **dncsatm** entry exists.

   Note: If the dncsatm entry does not already exist under the Hosts tab, contact Cisco Services.
3. Click the **Sources** tab; click **File** and then select **New**. The Set Up BFS Source window opens.

4 Use the example in the following illustration to configure the Set Up BFS Source window.

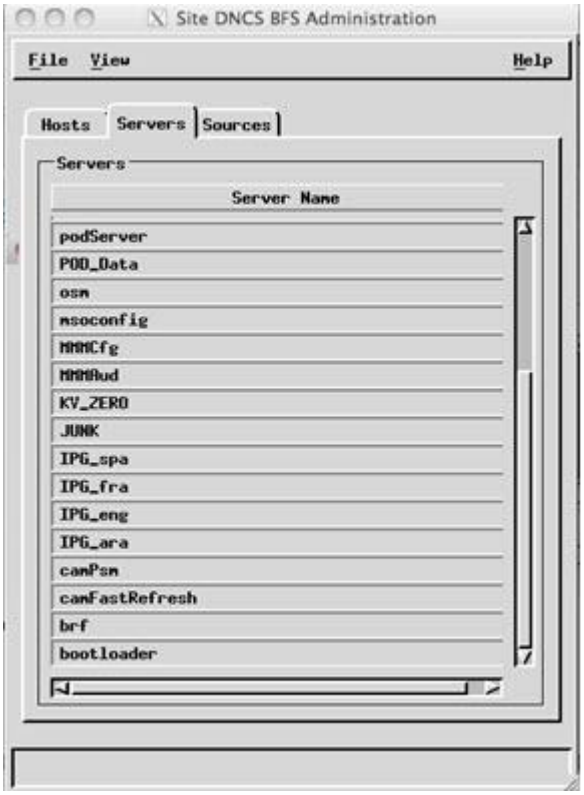
Notes:

- Use a value greater than 200 for the **Source ID**. Valued less than 200 are reserved for system sessions.
- **Data Rate** should be between 0.5 and 1.0 Mbps. The maximum allowable value is 2.0 Mbps.
- For **Block Size**, type **4000** for optimal performance.
- For **Available Hosts**, select **dncsatm** for a non-RCS system, or **AllSitesHost** for a RCS system.



5 Click **Save**.

6 Click the **Servers** tab.

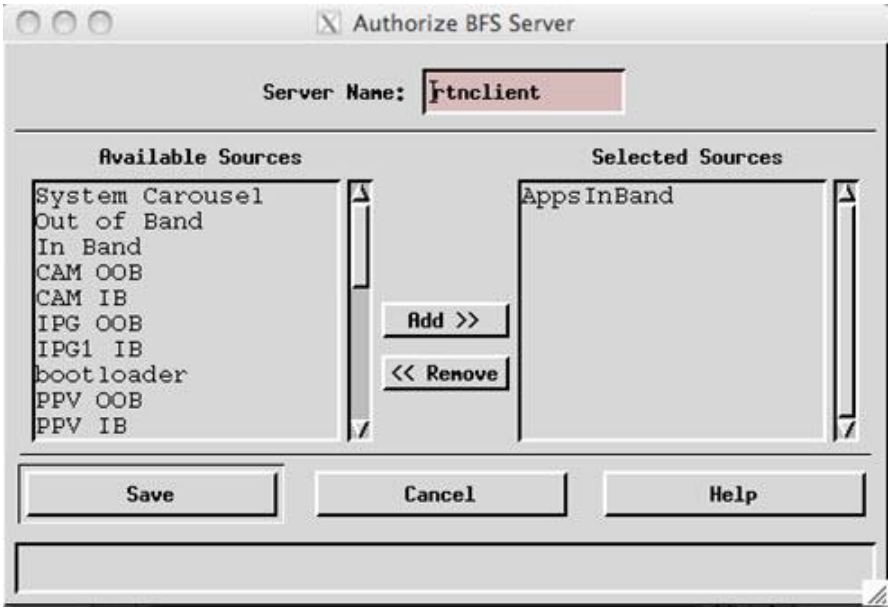


7 Click **File** and then select **New**. The Authorize BFS Server window opens.

8 In the **Server Name** field, type **rtnclient**.

9 From the **Available Sources** list, select the appropriate source and add it to the **Selected Sources** column.

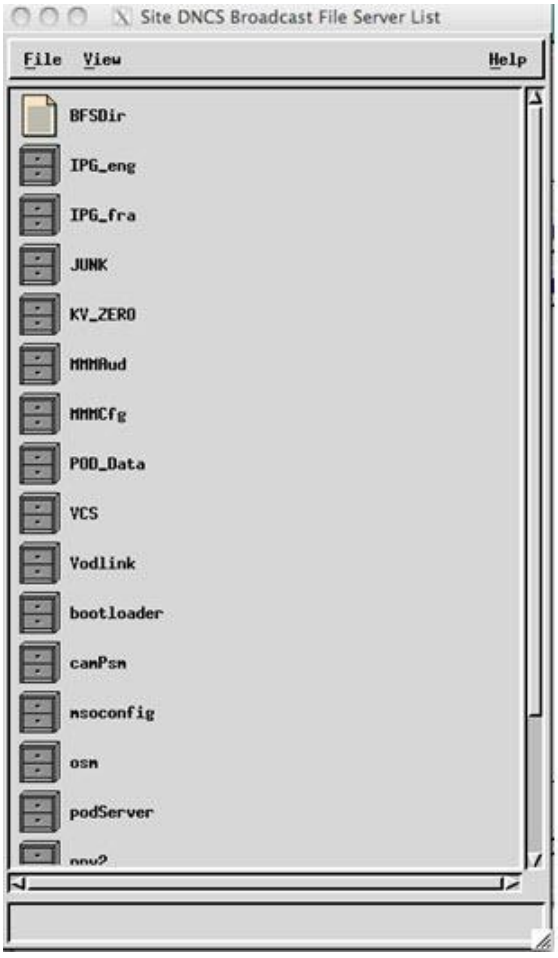
Note: In this example, it is **AppsInBand**.



10 Click **Save**.

11 On the DNCS Administrative Console, select the **Application Interface Module** tab.

12 Click **BFS Client**. The Site DNCS Broadcast File Server List window opens.



13 Click **File** and then select **New Server**. The Set Up Server window opens.

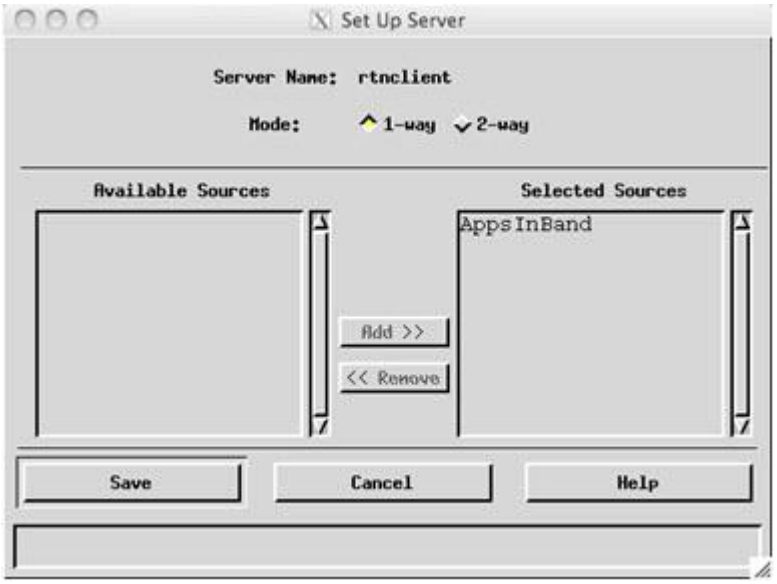
14 Complete these steps to configure the Set Up Server window.

a For **Server Name**, type **rtnclient**.

b For **Mode**, select **1-way**.

c In the **Available Sources** column, select the source you created for the rtnclient server and add it to the **Selected Sources** column.

Note: In the example from step 9, the source is **AppsInBand**.

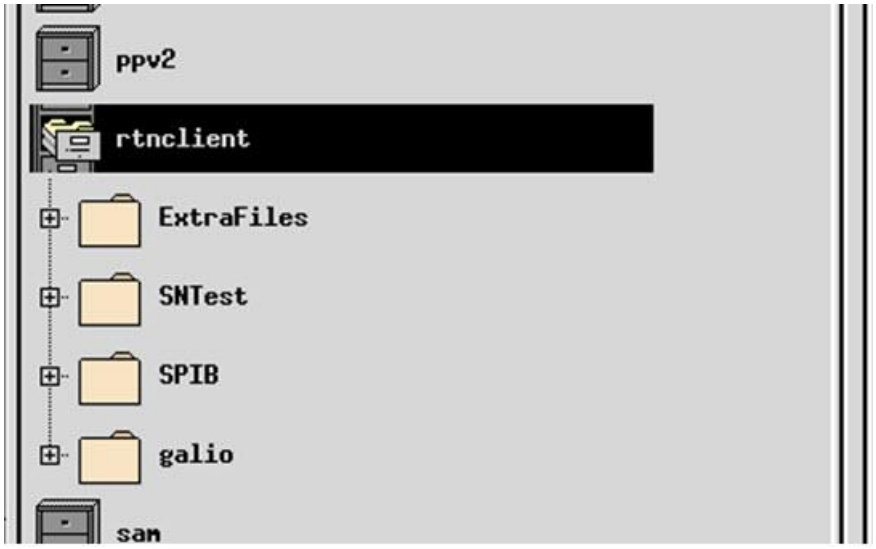


15 Click **Save**.

16 On the Site DNCS Broadcast File Server List window, select the newly created **rtnclient** server to highlight it.

17 Click **File** and then select **New Directory**. Name the new directory **galio**.

Important: The new directory must be named *galio*.



18 Select the newly created **galio** directory to highlight it.

19 Click **File** and then select **New Directory**. Name the new directory (a subdirectory under galio) **packages**.

Important: The new directory must be named *packages*.

20 Select the newly created **packages** directory to highlight it.

Note: All Push-downloadable objects must be added under the *packages* directory (DNCS Administrative Console > Application Interface Modules tab > BFS Client > rtnclient > galio > packages).

21 Click **File** and then select **New Link**. The Set Up Link window opens.

22 Follow these instructions to configure the Set Up Link window.

a For the **Link Name**, provide a name for your linked package and give it a *zip* extension.

Example: `ynws.zip`

b For the **Source Name**, select the source you created for the rtnclient server.

Example: `AppsInBand`

c For the **Linked Path**, locate the downloadable object you used in the **Link Name** field by clicking the **Select** button and then navigating to it.

Note: This assumes that you have already placed the object onto the DNCS.

23 Click **Save**.

Notes:

- Now that you have added your objects to the BFS carousel, a SAM service must be created for each of the objects that you wish to deploy. The object name must be configured in the SAM service according to the following example: _[package id] (for example, **`_ynws`**). Associate the SAM service with an EID (package identifier) as detailed in ***Create a SAM Service*** (on page 47).
- Refer to ***Assign the Entitlement Package to a Set-Top Box*** (on page 48) for instructions on how to authorize a set-top for the package.

The Pull Model for Deployment

The pull model of object deployment allows users to visit a web page on their set-top box that contains links to one or more objects. For example, a network operator may offer a portal where themes and widgets could be downloaded, or a games provider may offer access to games through their website.

In order to use the pull mode, the operator must first create an Application Store Walled Garden, using the Short Description **_WGAP**. The URL of the Application Store can be hosted on a locally managed server, or on a trusted server on the Internet.

The location of the Application Store must be added to the "white list" file, which is added to the variable in the dl-config.txt file. The white list is a list of websites from which Vantage can download objects. An attempt to download an object from any other site will result in an error. See *Cisco Videoscape Voyager Vantage Configuration Guide* (part number OL-26411) for additional details.

The name of the white list file is stored in the following dl-config.txt configuration variable:

```
mom.download.whitelist.file:
http://<server>/<location>/whitelist.txt
```

By default, the configuration variable is not set, which means that there are no restrictions from where packages can be downloaded. For security reasons, Cisco does not recommend having no restrictions from where packages can be downloaded. Cisco recommends the use of a white list file at all times.

If a file name is provided, but that file cannot be found, then no packages will be downloaded. If the file is found but it is empty, then no packages can be downloaded, as well.

The White List File

The white list file consists of a number of lines that terminate in a carriage return (LF or CRLF). Lines that begin with a hash symbol (#) are considered comments and are ignored. Blank lines are also ignored.

Lines in the white list file consist of a Uniform Resource Identifier (URI) fragment, followed by white space, and then followed by the string "allow".

The URI fragment follows these rules:

- If it starts with a dot ('.'), then it is matched against the right-hand side of the domain name
- If it does not start with a dot, then it is matched against the entire domain

- If it includes a port number, then the specific port must match; otherwise, any port is allowed
- If it includes one or more forward slashes ('/'), then it is prefix-matched against the path, as well as the domain

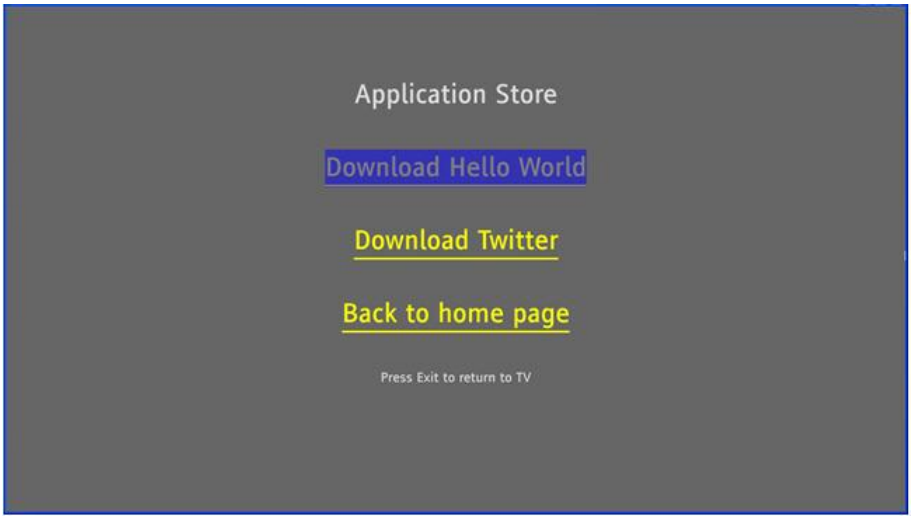
A sample white list file follows:

```
#Allow any sub-domain of cisco.com
.cisco.com allow
#Allow Anywhere on the apps.cisco.com server
apps.cisco.com allow
#Allow Any sub-directory of apps-v1 directory on the
apps.cisco.com server
apps.cisco.com/apps-v1 allow
```

The Walled Garden Application Store can instruct the STB to download objects using the javascript packageManager.fetch() function, as shown in the following example:

```
<html>
<head>
<title>Application Store</title>
<link rel="stylesheet" href="style.css">
<script>
function download(file)
{
  var basedir = document.location.href;
  basedir = basedir.substring(0, basedir.lastIndexOf('/')) + '/';
  packageManager.fetch(window, basedir + file)
}
</script>
</head>
<body>
<p class="title">Application Store</p>
<p class="link"><a
href="javascript:download('test.zip');">Download Hello
World</a></p>
<p class="link"><a
href="javascript:download('app_twitter.zip');">Download
Twitter</a></p>
<p class="hint">Press Exit to return to TV</p>
</body>
</html>
```

Subscribers can then access the Walled Garden Application Store from the Applications Menu and/or Guide (as determined by the walled garden configuration settings) to selectively download objects (applications, themes, language packs, and logos).



Just like with BFS-downloaded objects, the operator (either manually or using automated BOSS commands between the Walled Garden Application Store and the DNCS/billing system) must then authorize eligible STBs on the network for the package using the billing system with the SAM service name "_<object id>" parameter. See ***Provisioning STBs with Application, Theme, Logo, and Language Downloadable Objects*** (on page 43).

Note: Logo objects and upgrades to built-in objects, such as ipg, pvr, vod, and built-in themes, do not require authorization.

4

Provisioning STBs with Application, Theme, Logo, and Language Downloadable Objects

Introduction

This chapter describes the procedures used to provision STBs with the applications, themes, logos, and language packs created and deployed through the Object Download feature of Vantage.

In This Chapter

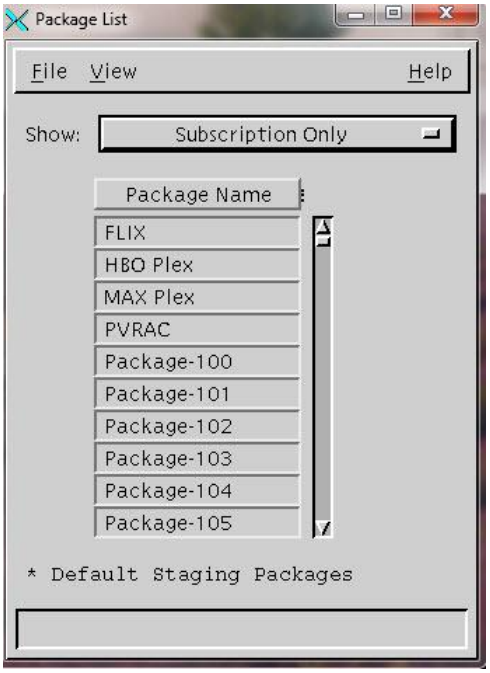
Create the Entitlement Package for the Object or Group of Objects

In this procedure, you will create entitlement packages. These steps need to be completed for each entitlement package you create.

Note: You can create one entitlement package per object (for example, one entitlement package for each application, theme, logo, or language object, such as the "App YNWS" package for authentication of the ynws object). Or, you can create an entitlement package for a group or suite of objects (for example, one entitlement package for multiple applications, such as an "Applications" package for authorization of multiple objects).

1 From the DNCS Administrative Console, click the **System Provisioning** tab.

2 Click **Package**. The Package List window opens.



3 Click **File** and then select **New**. The Set Up Package window opens.

4 In the **Package Name** field of the Set Up Package window, type the following:
`App YNWS`

5 Be sure the **Duration** is set to **Unlimited**.

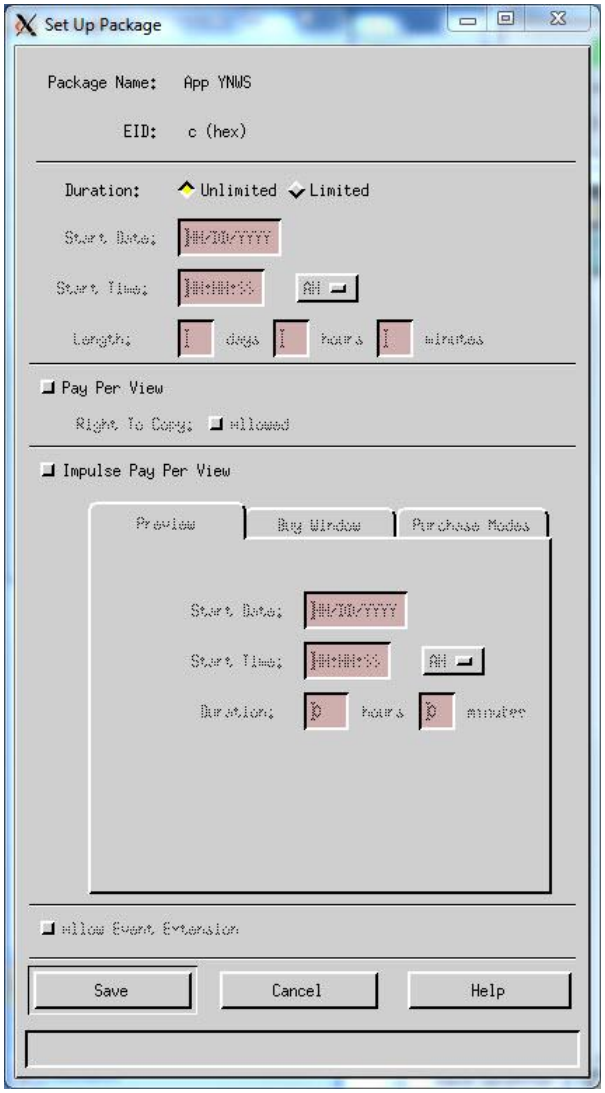
6 Click **Save**. The Set Up Package window closes.

7 From the Package List window, scroll down to find the just-created **App YNWS** package.

8 Highlight the **App YNWS** package.

9 Click **File** and then select **Open**. The Set Up Package window opens showing the just-created App YNWS package.

Example:



10 Take note of the Entitlement ID (**EID**) and record that value in the space provided:

_____

Example: In the example used in step 9, the EID is **c**.

11 The EID in step 9 (**c**) is in hexadecimal format. Access a hexadecimal-to-decimal conversion chart on the Internet, and then record the decimal equivalent in the space provided. The decimal equivalent is used in the SAM Service URL field in ***Create a SAM Service*** (on page 47).

_____

Example: The decimal equivalent of hexadecimal c is **12**.

12 Click **Cancel** to close the Set Up Package window.

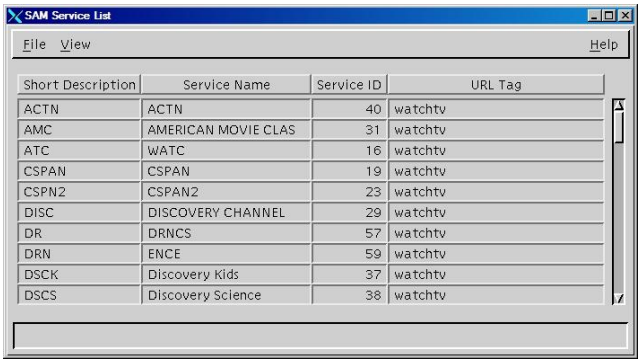
13 Click **File** and then select **Close** to close the Package List window.

Create a SAM Service

You need to create a SAM service for each object ID.

1 From the DNCS Administrative Console, click the **Application Interface Modules** tab.

2 Click **SAM Service**. The SAM Service List window opens.



3 On the SAM Service List window, click **File** and select **New**. The Set Up SAM Service List window opens.

4 Follow these instructions to configure the Set Up SAM Service List window.

a In the **Service Name** field, type **_ynws**.

b In the **Short Description** field, type **_ynws**.

c In the **Long Description** field, type **Yahoo! News App**.

d In the **Application URL** field, follow these guidelines:

– For Push-downloaded applications, the Application URL field should be:
`bfs:///rtnclient/galio/packages/[APP FILE ON BFS];eid=[EID from the App YNWS package]`
Example: `bfs:///rtnclient/galio/packages/ynws.zip;eid=12`

– For Pull-downloaded applications, the Application URL field should be:
`DummyURL;EID=[EID from the App YNWS package]`
Example: `DummyURL;EID=12`

Notes:

– Substitute for **[EID from the App YNWS package]** the *decimal* EID that you recorded in step 11 of ***Create the Entitlement Package for the Object or Group of Objects*** (on page 44).

– Leave all other fields on the Set Up SAM Service List window at their default values.

e Click **Save**. The Set Up SAM Service List window closes.

5 Click **File** and select **Close** to close the SAM Service List window.

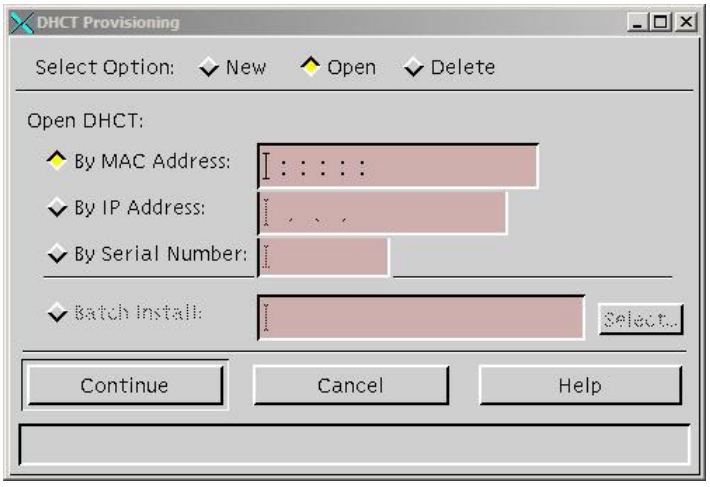
Assign the Entitlement Package to a Set-Top Box

In this procedure, you will assign an entitlement package to an STB.

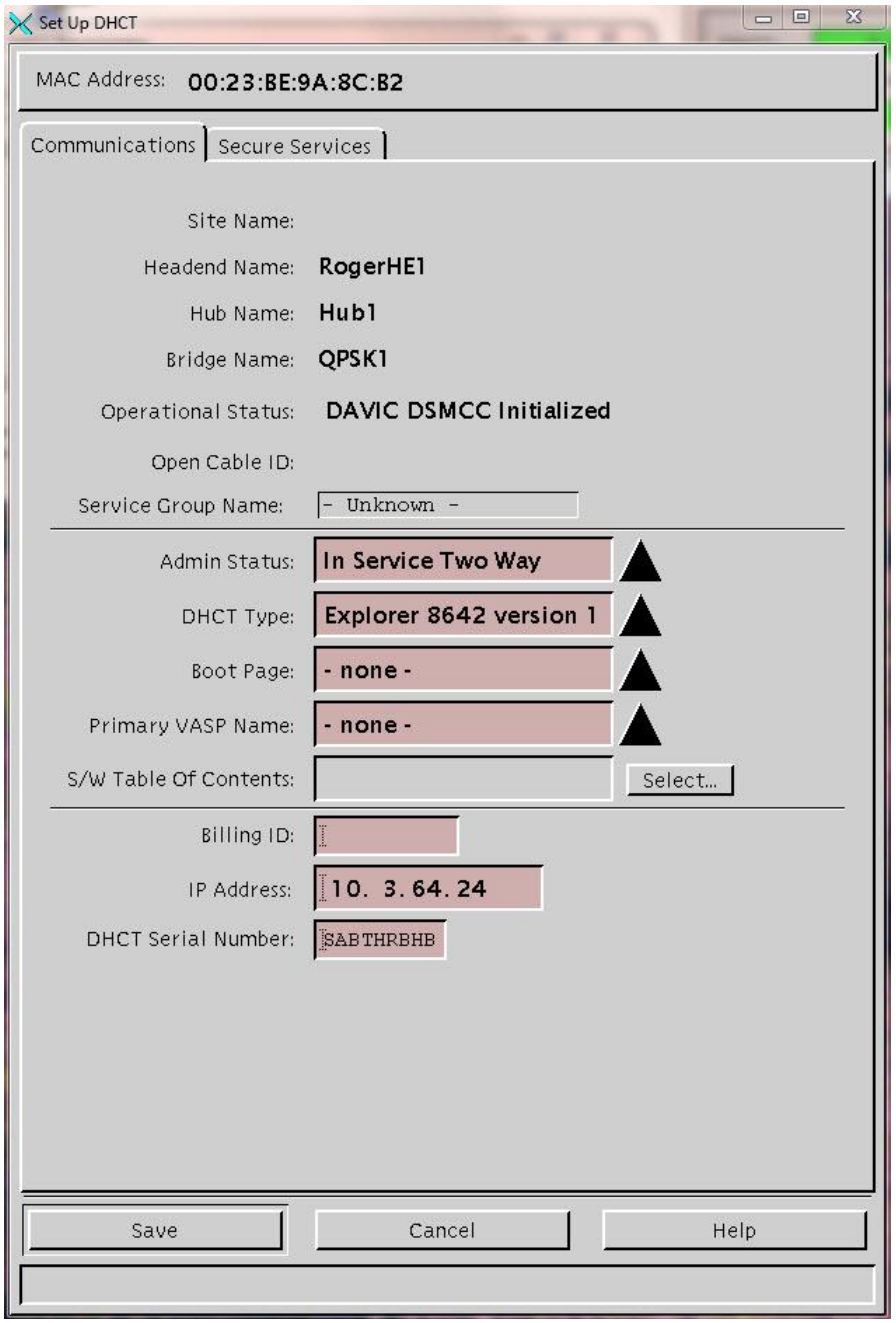
Note: Be sure that the STB is provisioned with Entitlement Management Messages (EMMs) for package authorization.

1 From the DNCS Administrative Console, click the **Home Element Provisioning** tab.

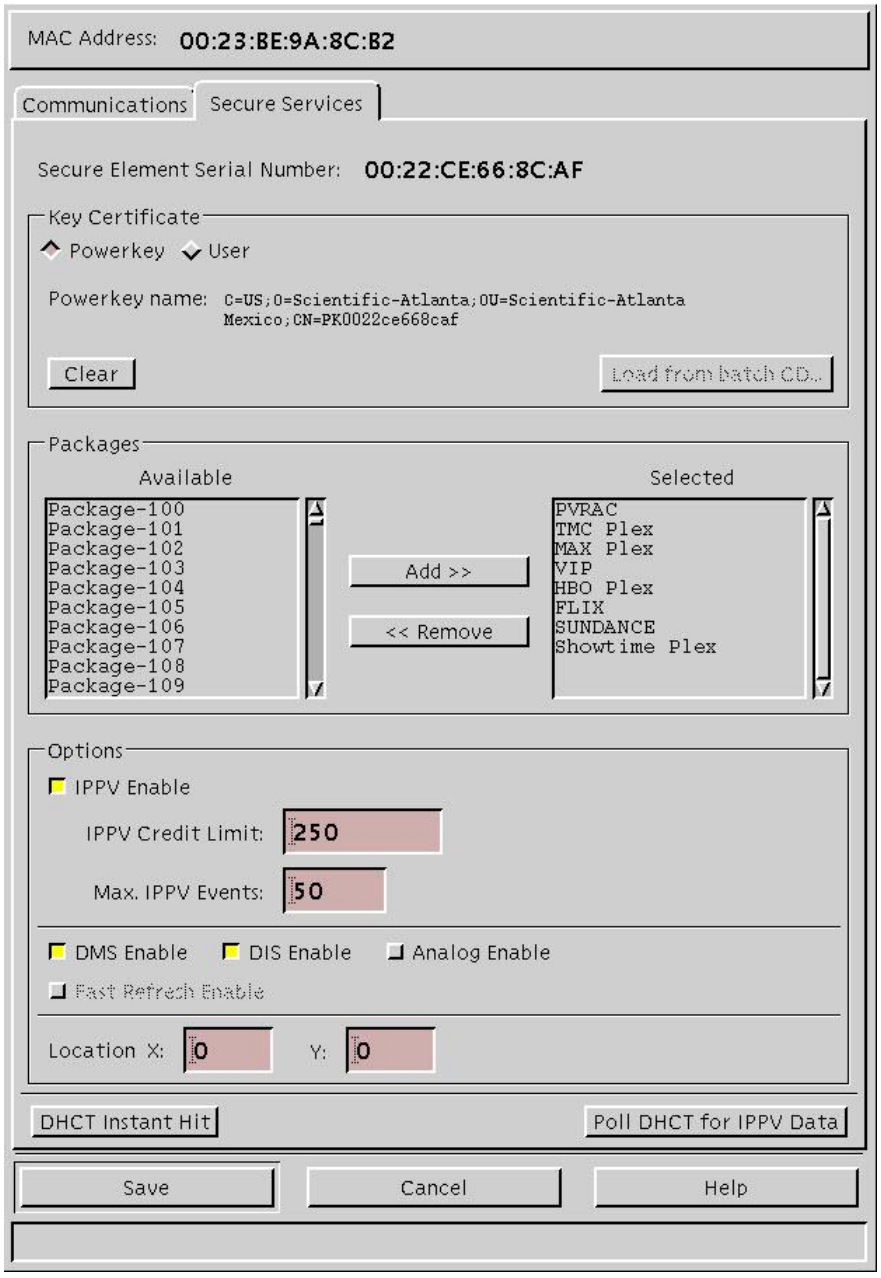
2 Click **DHCT**. The DHCT Provisioning window opens.



3 Click **By MAC Address** and type the MAC address of a DHCT in the test lab, and then click **Continue**. The Set Up DHCT window opens.



4 Click the **Secure Services** tab. The window updates to allow you to assign a package or packages to the DHCT.



5 Locate and then highlight the **App YNWS** package in the **Available** column of the **Packages** section of the window.

6 Click **Add**. The App YNWS package moves from the **Available** column to the **Selected** Column.

7 Click **Save**.

8 Click **Close** to close the Set Up DHCT window.

9 Click **Cancel** to close the DHCT Provisioning window.

5

Configuring and Authorizing Walled Gardens on a Set-Top

Introduction

The signaling from the DNCS to provision Walled Garden services is accomplished by defining Walled Garden SAM service definitions on the DNCS, one for each Walled Garden service that is to be provisioned.

You can define up to 10 generic Walled Garden services for use on Vantage. Additionally, two special Walled Garden services for "My Account" and "App Store" can be defined.

In This Chapter

Walled Garden SAM Service Settings

Use the following fields when you create and manage Walled Garden services on the DNCS.

Field	Description
Service Name	Required field The name of the service is only available and shown to users of the DNCS who have access to the SAM Services List. Make this descriptive enough to understand what Walled Garden service is being offered. Subscribers will not see this field. **Example: `Walled Garden – App Store`**
Short Description	Required field The Short Description is from a set of well-known values defined in the following list. The set-top uses these short description values to determine whether those services should be displayed on the Vantage UI. If you are configuring the Walled Garden service to appear in the EPG, this field is used as the channel indicator. **Example: _WG00** The following short description names are reserved for provisioning Walled Garden services. Each of these short descriptions must only be used once per system and must be used *exactly* as shown. ■ **_WG00** - Main service for Walled Garden ■ **_WG01** - Service for Walled Garden 01 ■ **_WG02** - Service for Walled Garden 02 ■ **_WG03** - Service for Walled Garden 03 ■ **_WG04** - Service for Walled Garden 04 ■ **_WG05** - Service for Walled Garden 05 ■ **_WG06** - Service for Walled Garden 06 ■ **_WG07** - Service for Walled Garden 07 ■ **_WG08** - Service for Walled Garden 08 ■ **_WG09** - Service for Walled Garden 09 ■ **_WGMA** - Service for Walled Garden "My Account" ■ **_WGAP** - Service for Walled Garden "App Store"
Long Description	The Long Description is for meaningful display to the subscriber. This field appears in the menu and/or in the grid cell of the EPG, as indicated by the options in the Application URL field, and is up to you. **Example: Add Applications**

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>Application URL</td><td>The Application URL provides additional information for how the Walled Garden service can be accessed. It contains the application name and additional attributes, separated by semicolons.

■ inEPG
– 0 (zero) — Service is not displayed in the EPG
– 1 (or name) — Service is displayed in the EPG (default)
Note: The inEPG flag is independent of whether the service is channel-mapped or not.

<table>
<tr><th>Channel Mapped</th><th>inEPG Flag</th><th>Result</th></tr>
<tr><td>No</td><td>X</td><td>Not listed in EPG</td></tr>
<tr><td>Yes</td><td>0</td><td>Not listed in EPG; Direct tune brings up the service</td></tr>
<tr><td>Yes</td><td>1</td><td>Listed in EPG; Direct tune brings up the service</td></tr>
</table>
inMenu
– 0 (or name not present) — Service is not displayed in the Applications Menu (default)
– 1 (or name is present) — Service is displayed in the Applications Menu

■ eid — The authorization EID (decimal) of the service. This is to ensure that you can authorize this Walled Garden service to individual STBs. See Create the Walled Garden Package (on page 57) for details on creating a Walled Garden package and authorizing a set-top for the package.

■ url — The URL to access the service. This must be the last attribute of the Application URL so that it is not required to be escaped

Format:
<tt>wgarden://wgarden;inEPG=0;inMenu=1;eid=[xx];url=http://[server]/[location]</tt>

Notes:

■ Replace [xx] with the EID (decimal) value associated with the Walled Garden package. See Create the Walled Garden Package (on page 57).

■ Replace [server] with the name or IP address of the server.

■ Replace [location] with the location of the web page/application.

■ The syntax requires the EID to be placed before the URL.</td></tr>
</table>

Field	Description
Logo	The channel or application logo number associated with this walled garden application **Note:** The logo functions in the EPG in the same way as current channel logos.
Parameter	Select the **Number** option
Number	Enter **0** (zero)

Add a Walled Garden SAM Service

Complete these steps to add a new Walled Garden service.

1. From the DNCS Administrative Console, select the **Application Interface Modules** tab.
2. Click **SAM Service**. The SAM Service List window opens.
3. Click **File** and then select **New**. The Set Up SAM Service window opens.
4. Complete the fields on the Set Up SAM Service window, as described in ***Walled Garden SAM Service Settings*** (on page 52).
5. Save the Set Up SAM Service window when you are finished.

Modify the SAM Services List

Complete these steps to modify an existing Walled Garden service.

1. From the DNCS Administrative Console, select the **Application Interface Modules** tab.
2. Click **SAM Service**. The SAM Service List window opens.
3. Locate and highlight the service you want to modify.
4. Click **File** and then select **Open**. The configuration window for the service selected opens.
5. Edit the fields as described in ***Walled Garden SAM Service Settings*** (on page 52).
6. Click **Save**.
7. If prompted, confirm the changes.

Create the Walled Garden Package

Complete these steps to create a Walled Garden authorization package.

1. From the DNCS Administrative Console, click the **System Provisioning** tab.
2. Click **Package**. The Package List window opens.
3. Click **File** and then select **New**. The Set Up Package window opens.
4. In the **Package Name** field of the Set Up Package window, type the following:
   `Walled Gardens` (or `Walled Garden 00` if you want to assign a package for each Walled Garden service)
5. Be sure the **Duration** field is set to **Unlimited**.
6. Click **Save**. The Set Up Package window closes.
7. From the Package List window, scroll down to find the **Walled Gardens** package you just created.
8. Highlight the **Walled Gardens** package.
9. Click **File** and then select **Open**. The Set Up Package window opens showing the just-created Walled Gardens package.
10. Take note of the Entitlement ID (**EID**) (in both hexadecimal and decimal form) and record those values.
    hex: _____ decimal: _____
11. Click **Cancel** to close the Set Up Package window.
12. Click **File** and then select **Close** to close the Package List window.

Assign the Walled Garden Package to the Set-Top

In this procedure, you will assign the Walled Garden package to a set-top box.

1. From the DNCS Administrative Console, click the **Home Element Provisioning** tab.
2. Click **DHCT**. The DHCT Provisioning window opens.
3. Click **By MAC Address** and type the MAC address of a DHCT you wish to authorize, and then click **Continue**. The Set Up DHCT window opens.
4. Click the **Secure Services** tab. The window updates to allow you to assign a package or packages to the DHCT.
5. Locate and then highlight the **Walled Garden** package in the **Available** column of the **Packages** section of the window.
6. Click **Add**. The Walled Garden package moves from the **Available** column to the **Selected** Column.
7. Click **Save**.
8. Click **Close** to close the Set Up DHCT window.
9. Click **Cancel** to close the DHCT Provisioning window.
10. Reboot the set-top box to see the object immediately.

Cisco Systems, Inc.
5030 Sugarloaf Parkway, Box 465447
Lawrenceville, GA 30042

678 277-1120
800 722-2009
www.cisco.com

February 2012 Printed in United States of America Part Number OL-26414-01