Configuring Downloadable Objects in an ISDP Network

Application Guide

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 IPTV Services Delivery Platform (ISDP). Systems that use the Object Download feature of the ISDP must currently support version 2.6 or later of client 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 the ISDP.

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 ISDS 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 ISDS system expertise
- The ability to add and remove user accounts

Document Version

This is the second formal release of this document.

1

Overview of the Object Download Feature

Introduction

The Object Download feature of ISDP version 2.6 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 defines 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 widgets, 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 or channel
- **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 ISDS.
- Subscribers download these items at will (pull), via 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.

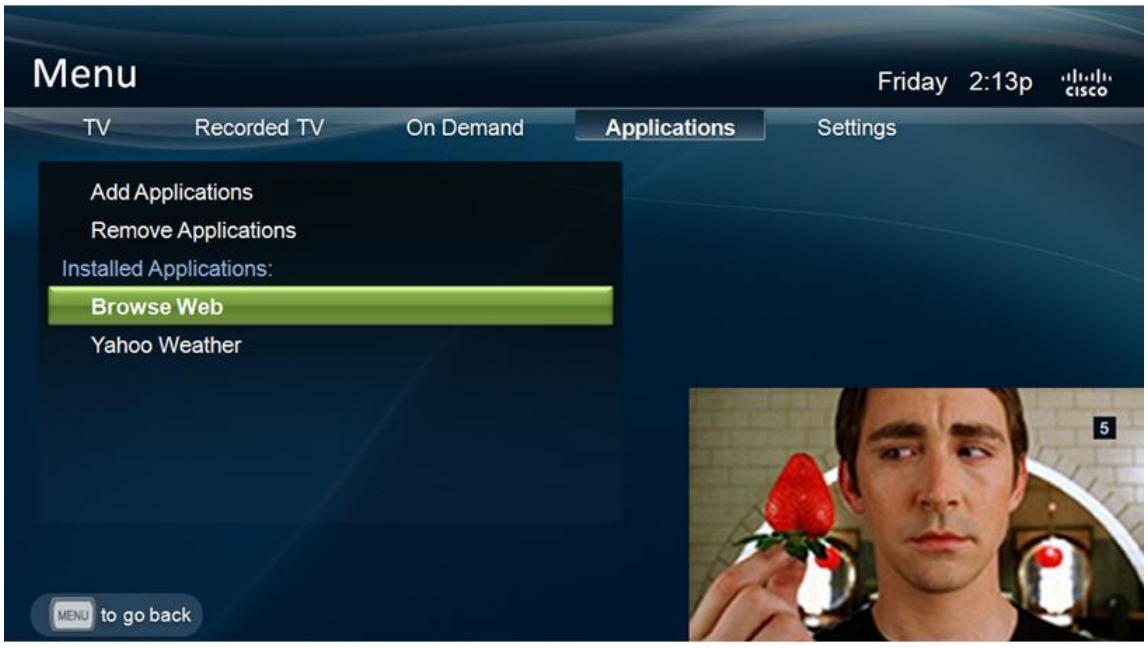
Both of these methods require configuration on the ISDS.

The Subscriber Interface

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

Main Menu

The **Applications** menu item replaces the **Services** menu.



Notes:

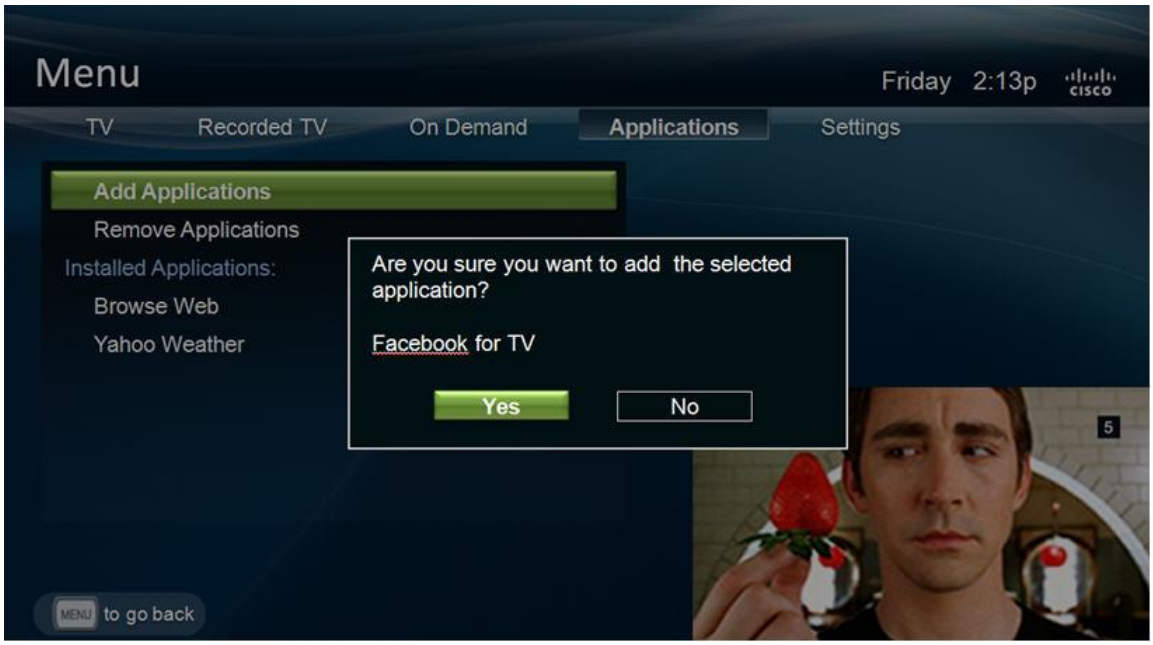
- The **Installed Applications** menu item represents the current list of installed (push or pull) applications and provisioned Walled Gardens that are configured to appear in the Applications menu (through the config.xml file).
- Selection of **Add Applications** leads to the Application Store provisioned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

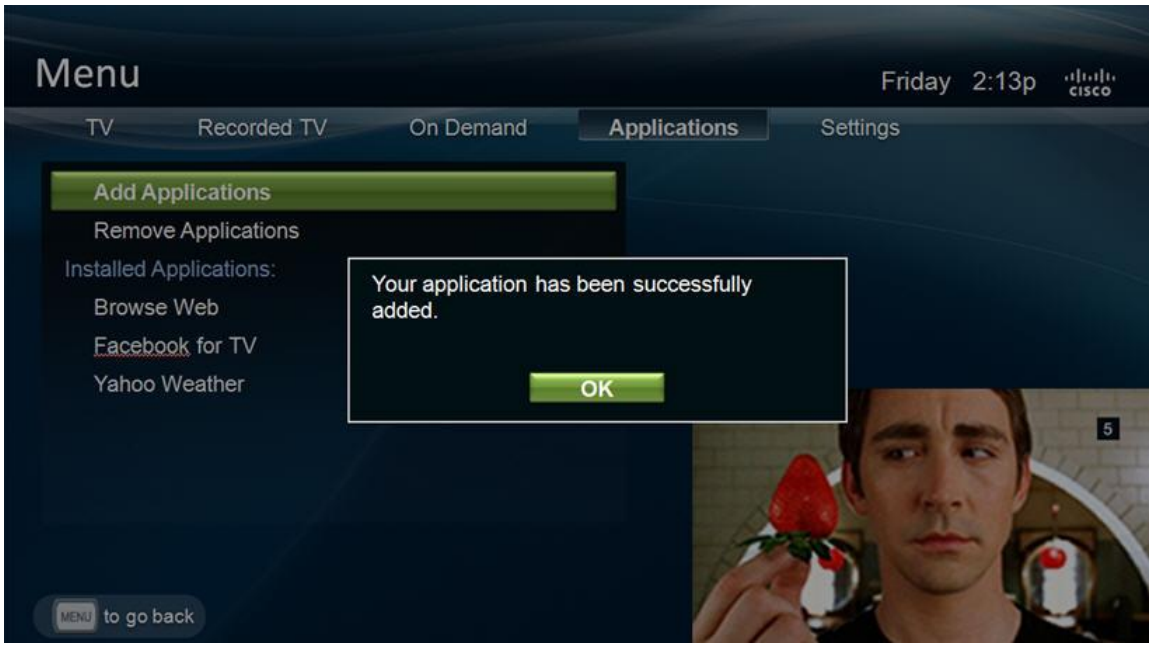
Available Applications (Through the Application Store)

The following image is an example of the Walled Garden Application Store. The subscriber selects an application to install from the list.

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.

2

Creating the config.xml Configuration File

Introduction

This chapter helps software developers accomplish the following tasks needed to implement the Object Download feature:

- 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 a package in order for the package to be valid. It is always called config.xml and always resides in the root of the package 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>
```

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

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

Attributes

<package id>

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

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

<type>

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

The localized name of the package 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>

Localized long description of the package as 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>

Author/contact details for package. Free text format. Can be name, email address, etc. The author element is not displayed on the user interface.

<version>

This is a string that will be used to display the package version to the user but will also be compared against other version strings to determine whether a new package 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 package. Package ids are assumed to be unique within an organization. Therefore, an id and orgId together provide a globally unique package name. This should be a domain style name, such as com.antplc or com.cisco.isdp.

<galioVersion>

Galio is the STB web browser used in conjunction with the Object Download feature of the ISDP. 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 that currently in use are not visible to the UI and are treated as if they do not exist.

<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 a theme provides. (Usually "1280x720" or "640x480")
lowMemProtect	yes/no	no	Should the application be protected from being killed to free memory when RTN 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 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 below 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 packages. 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 package, 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 package, 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 package 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 reliable.

<feature>

By using the feature element, a package declares that it uses this feature. If the required attribute is set to "yes" then the package cannot be run unless that feature is present. If the feature is not present, then the package is ignored and is not presented in the package list. 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 package.

Developing Packages

Application Packages

The following demonstrates the layout of an application package. 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 package or in language/theme packages.

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 package contains all the necessary library files for the theme. It can optionally contain application-specific theme files as well. Multiple resolutions for a theme are considered to be just parts of the theme. For example, you cannot download two packages, MyTheme-SD and MyTheme-HD. They must be included in the same package.

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

The following demonstrates the layout of a theme package.

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

Logo Packages

A logo package typically contains operator logos that are displayed on applications for branding. The logos may be customized for particular themes or may be generic and useable on any theme. Having them in a separate package allows simple customization of one theme for multiple operators without having to rebuild the theme package (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 package 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 package 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 package for the current language. If not found there, then the system searches the package 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 package. If not found there, then the system searches in the <app>/css/themes/<theme> directories of the current language package. Finally, the current theme package 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/js/messages/<lang>.js in the current language package. The application dictionary is loaded from js/messages/<lang>.js in the application package or, if not found, <app>/js/messages/<lang>.js in the current language package.

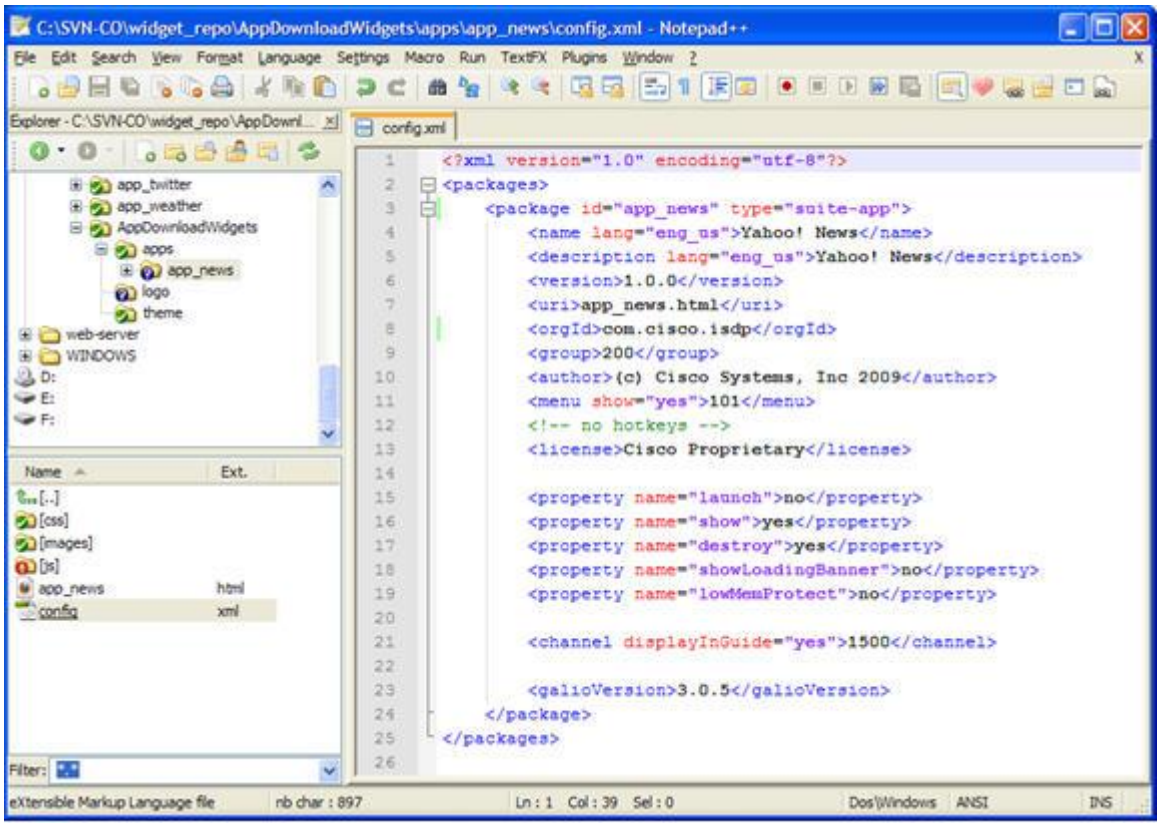
- <app> - current application
- <lang> - current language
- <www> - current screen width in pixels
- <hhh> - current screen height in pixels

Bundling Packages

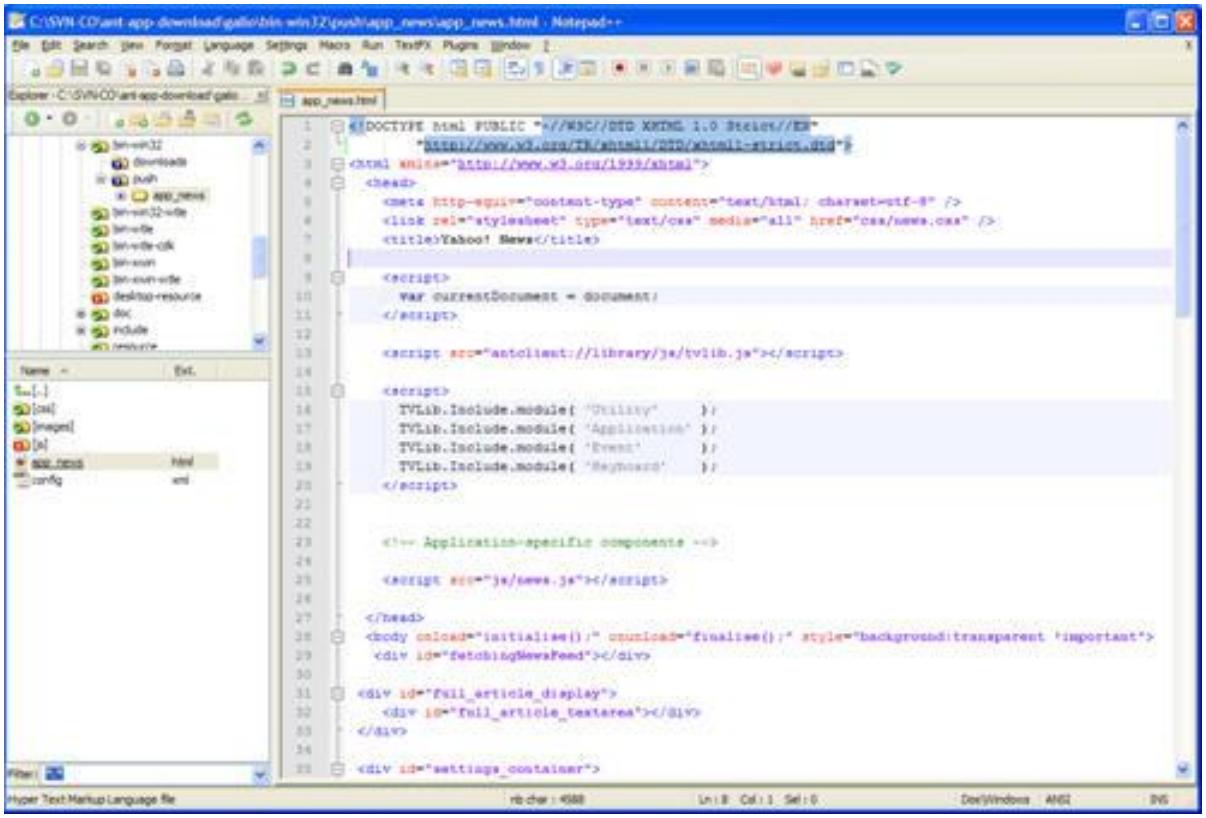
Once the config.xml file and the packages have been developed, the developer can go about the steps of bundling the package to download to the ISDS through the BFS carousel (push) and/or HTTP (pull).

Bundling Applications

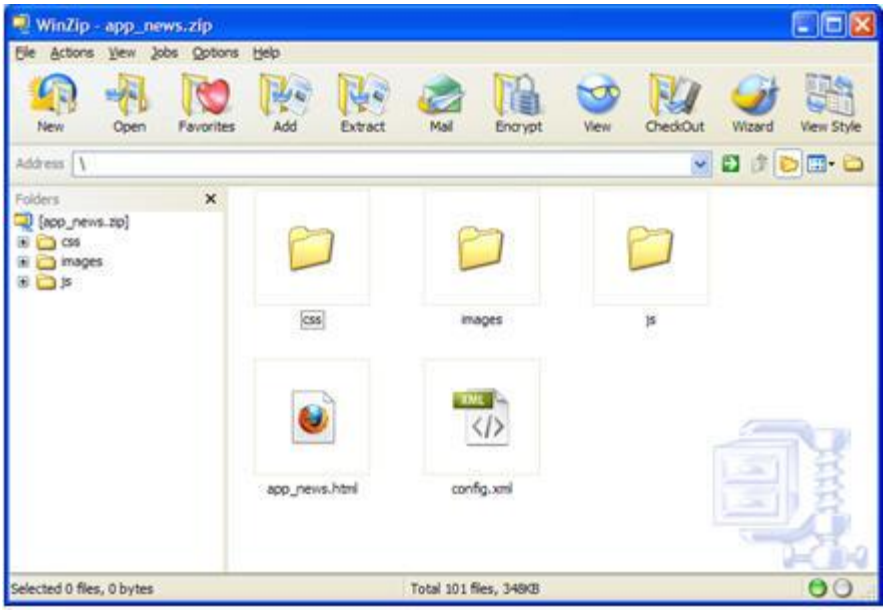
The accompanying screenshot displays the "app_news" application and its corresponding config.xml file. In this application, the app_news.html file is the initial launch file and the config.xml file is included with settings as described earlier in this chapter. This application will appear at location 101 in the Services menu (just under the built-in applications) and as channel 1500 in the 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 RTN 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 (RTN) modules. For example, you can see the **Utility**, **Event**, **Application**, and **Keyboard** modules.

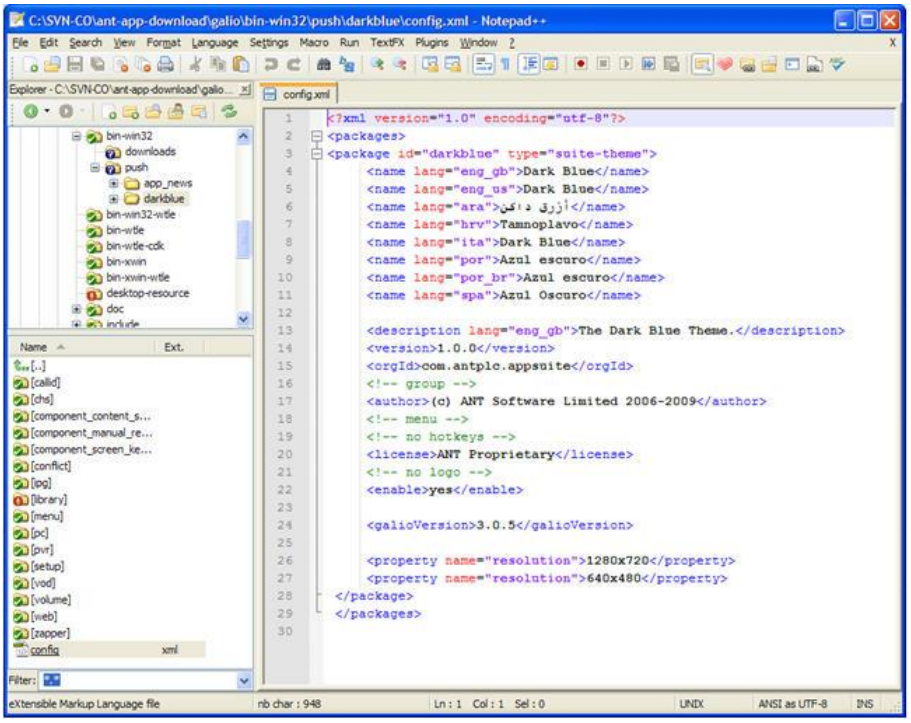


This package can then be compressed (zipped) and added to the Broadcast File System (BFS) (for push) or a walled garden server (for pull) for distribution to subscribers. After compression, the app_news.zip file might look like the following example:

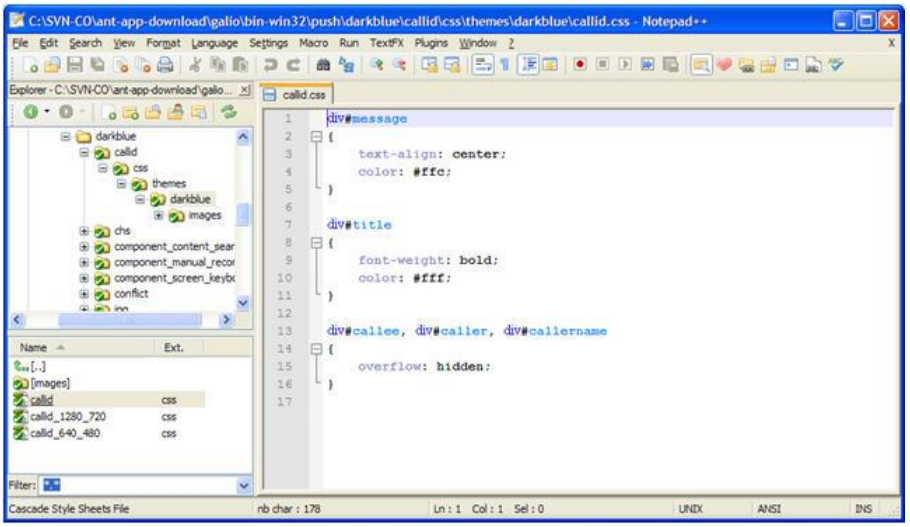


Bundling Themes

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.

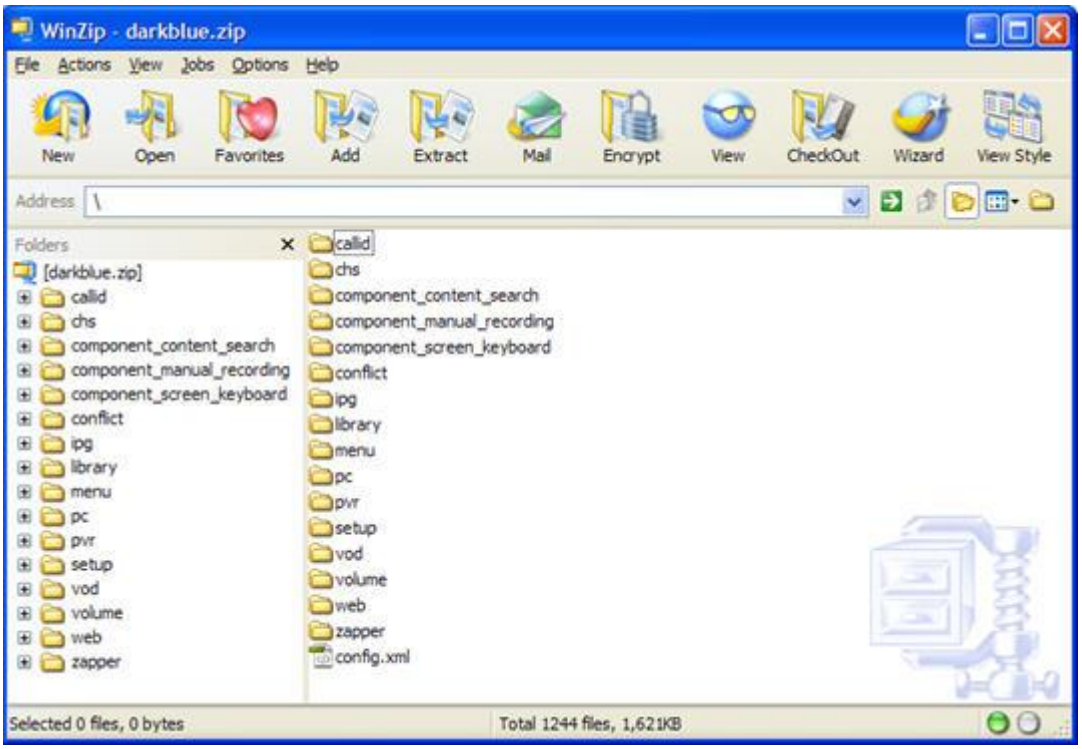


Notice the expanded **callid** application theme defined in the following example:



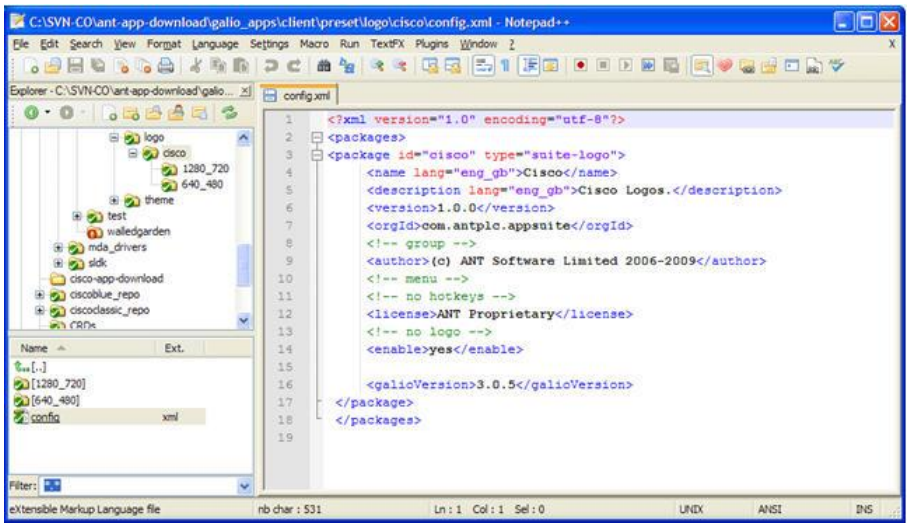
The darkblue theme for the callid application contains the following files: callid_1280_720.css, callid_640_480.css, callid.css. An image folder contains the image assets for the callid application using the darkblue theme.

This package 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:

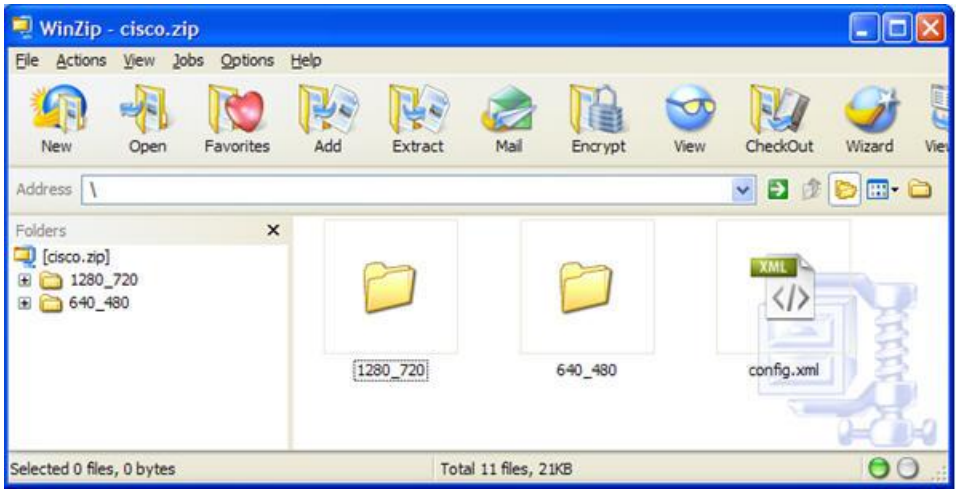


Bundling Logo Packs

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 operators logo in each directory.

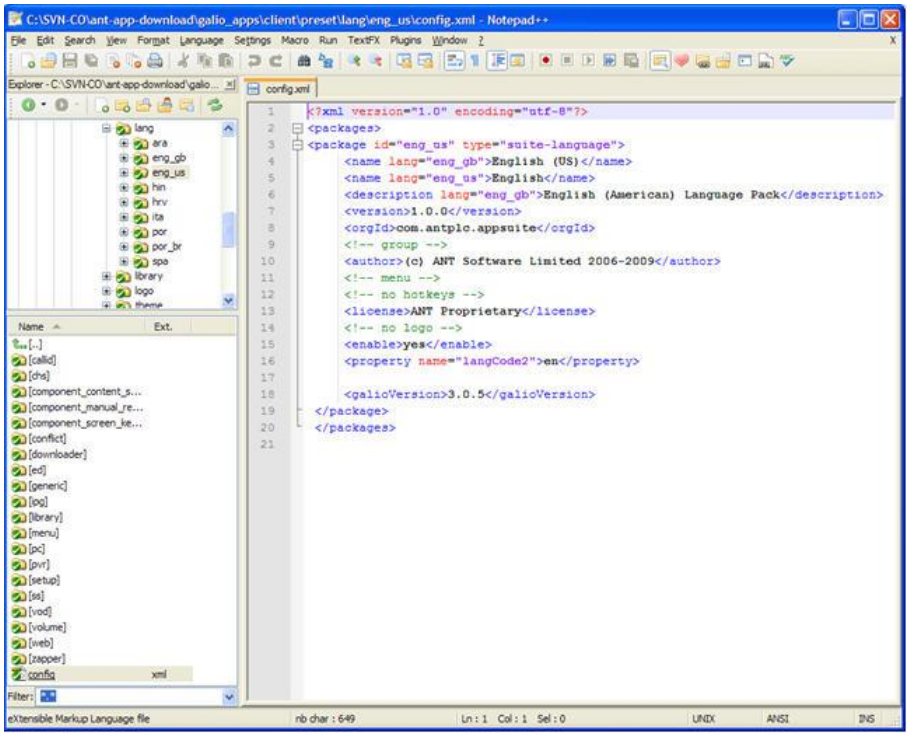


This package can be compressed (zipped) and added to the BFS or walled garden web server for distribution to subscribers. The cisco.zip file might look like the following example:

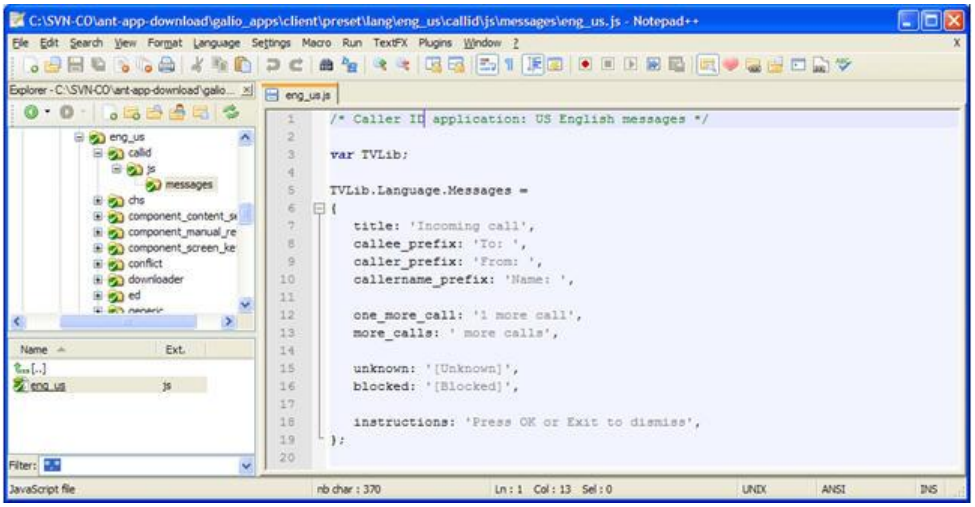


Bundling Language Packs

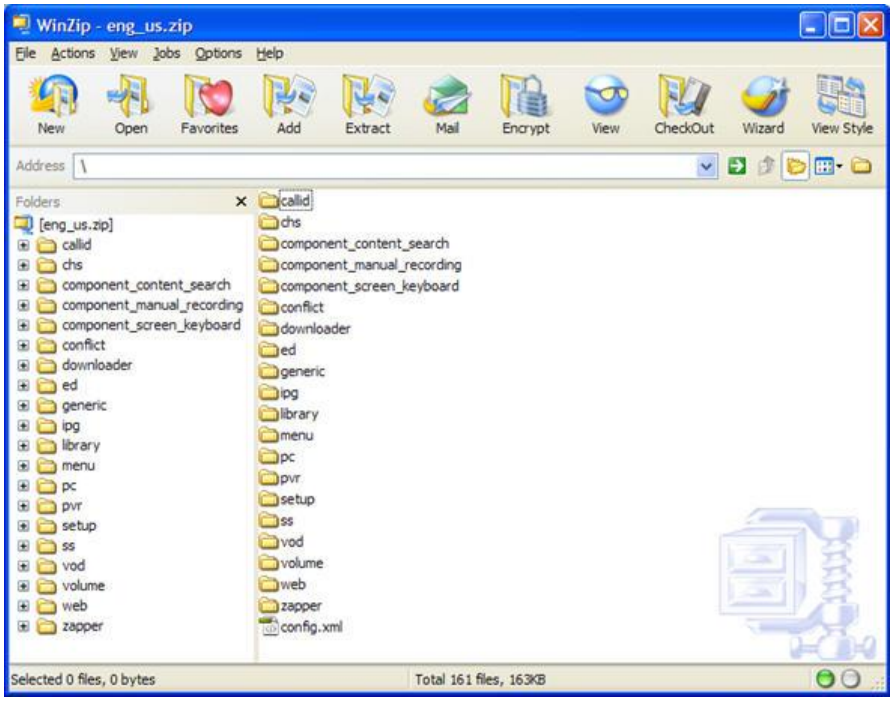
In the following example, notice the American English Language Pack and its associated config.xml file. With language packs, each language is its own package, and, just like with theme packs, each application has its own sub-directory. In each application sub-directory, there are js (javascript) and messages 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 package 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 packages, 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, Galio, the STB web browser, checks the following two locations for downloaded packages:

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

  Note: The default BFS location for placing packages is **/bfs/isdpclient/galio/packages**.

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

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

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

The Push Model for Deployment

The push model of package deployment allows an operator to send packages 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 package into the BFS directory at /bfs/isdpclient/galio/packages/.

Example: `/bfs/isdpclient/galio/packages/app_news.zip`

The operator must then authorize eligible STBs on the network for the package using the billing system and the service name "_<package id>" (e.g. "ynews").

Example: `_ynews`

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

See *ISDP System Release 2.6 Release Notes* (part number 4021187) for information on how to authorize services, including objects. Also, see ***Provisioning STBs with Application, Theme, Logo, and Language Packages*** (on page 37).

The Pull Model for Deployment

The pull model of package deployment allows users to visit a web page containing links to one or more packages. 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 the package downloader can download packages. An attempt to download a package from any other site will result in an error.

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

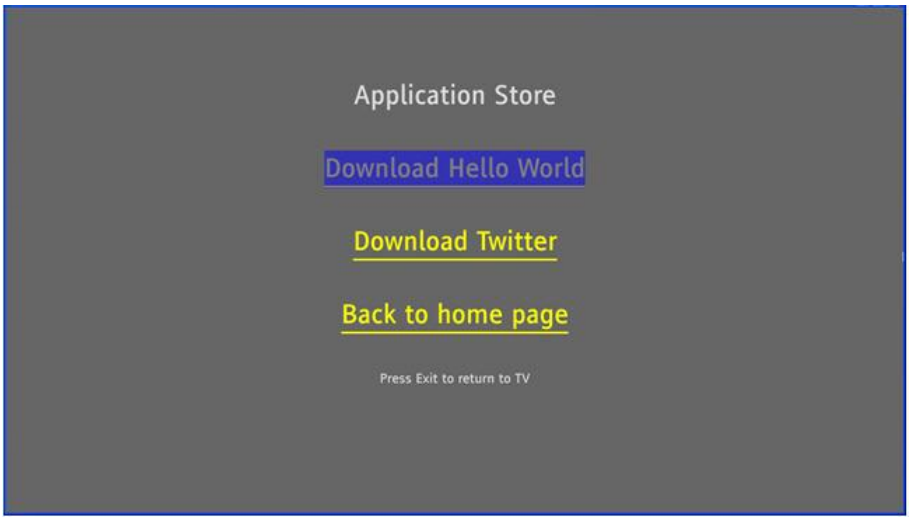
A sample white list file follows:

```
#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 packages 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 Services Menu and/or Guide (as determined by the walled garden configuration settings) to selectively download packages (applications, themes, language packs, and logos).



Just like with BFS-downloaded packages, the operator (either manually or using automated BOSS commands between the Walled Garden Application Store and the ISDS/billing system) must then authorize eligible STBs on their network for the package using the billing system with package name "_<package id>" parameter. See *ISDP System Release 2.6 Release Notes* (part number 4021187) for information on how to authorize services, including objects. Also, see ***Provisioning STBs with Application, Theme, Logo, and Language Packages*** (on page 37).

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

4

Provisioning STBs with Application, Theme, Logo, and Language Packages

Introduction

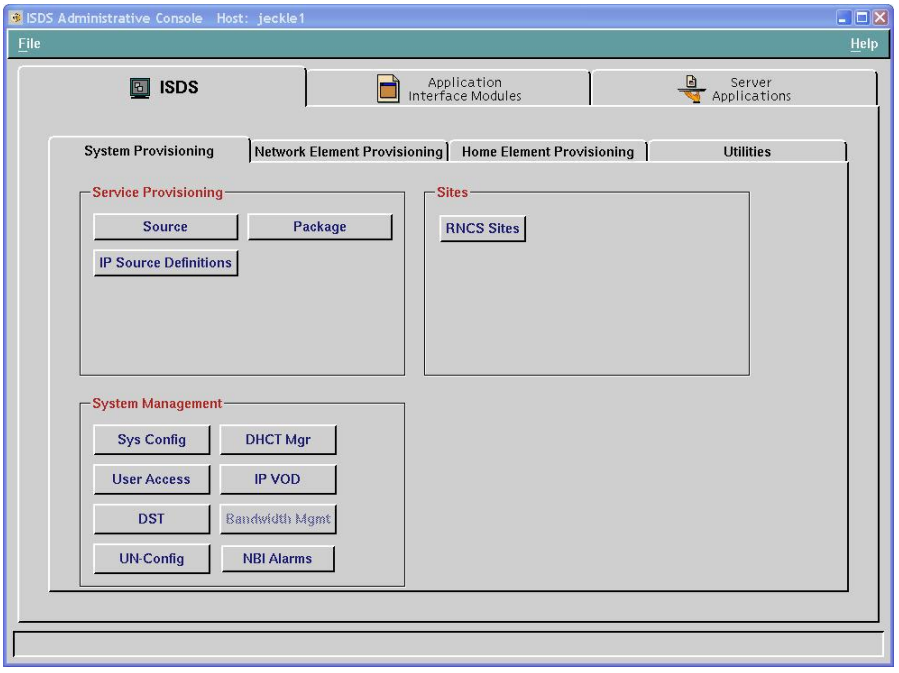
This chapter describes the procedures used to provision STBs with the applications, themes, logos, and language packs created and deployed through the Object Download feature of the ISDP.

In This Chapter

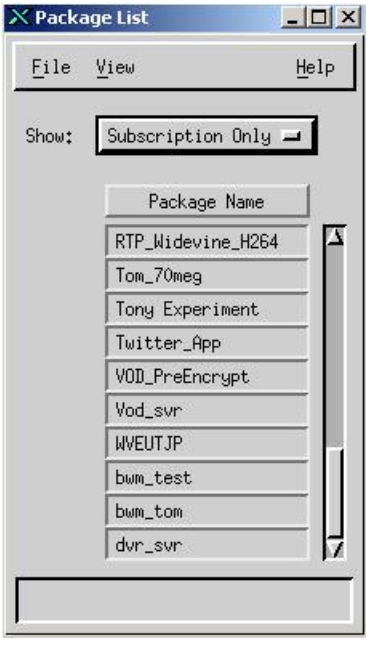
Create Application, Theme, Logo, and Language Packages on the ISDS

Follow these instructions to provision (or authorize) an STB for the applications, themes, logos, and language packages downloaded through the Object Download feature of the ISDS.

1 On the ISDS Administrative Console, click the **System Provisioning** tab.

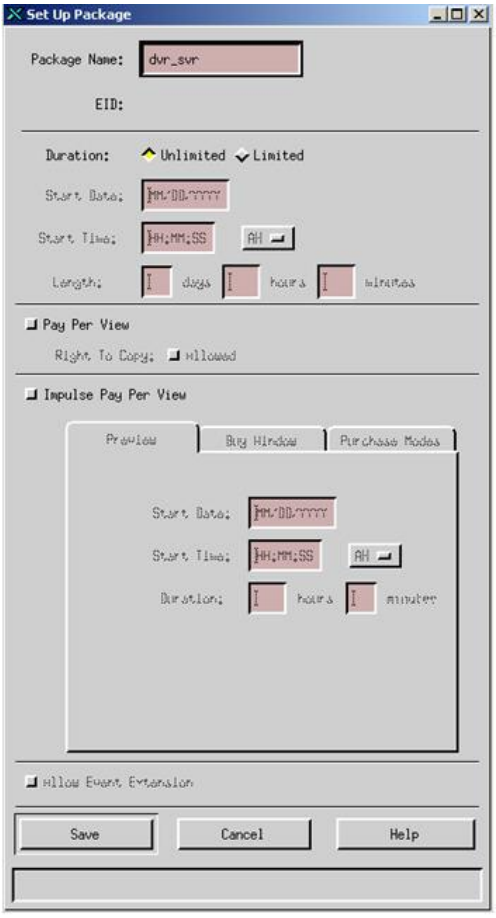


2 Click **Package**. The Package List window opens.



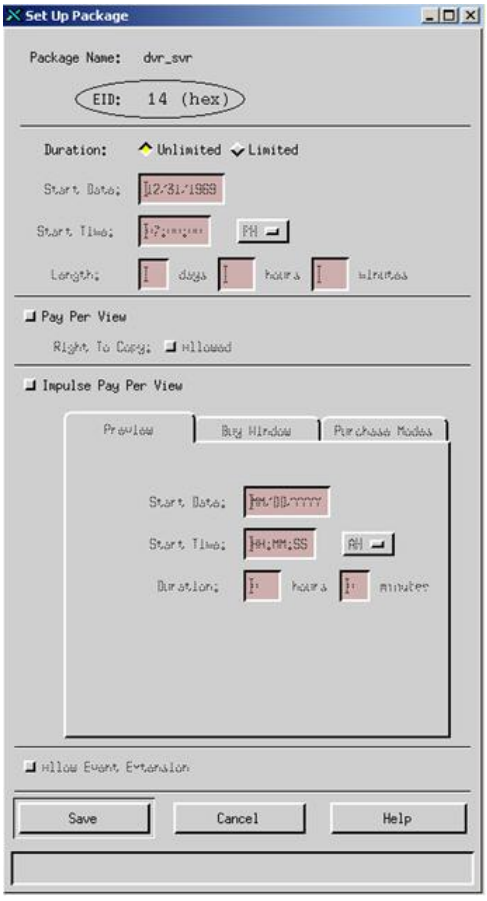
3 Click **File** and then select **New**. the Set Up Package window opens.

4 In the **Package Name** field, type the name of the package that you would like to assign.



5 Click **Save**.

6 Reopen the Set Up Package window and note the **EID** value.

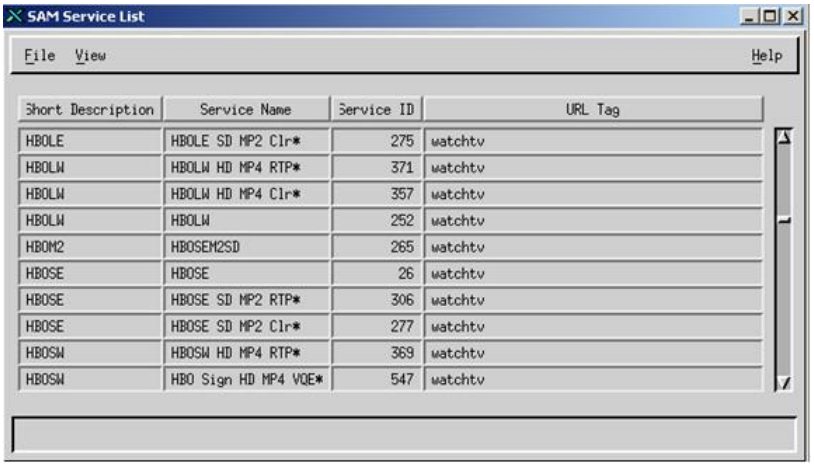


Note: In this example, the EID is **14**.

7 Close the Set Up Package window.

8 On the ISDS Administrative Console, select the **Application Interface Module** tab.

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



Short Description	Service Name	Service ID	URL Tag
HBOLE	HBOLE SD MP2 Clr*	275	watchtv
HBOLW	HBOLW HD MP4 RTP*	371	watchtv
HBOLW	HBOLW HD MP4 Clr*	357	watchtv
HBOLW	HBOLW	252	watchtv
HBOM2	HBOSEM2SD	265	watchtv
HBOSE	HBOSE	26	watchtv
HBOSE	HBOSE SD MP2 RTP*	306	watchtv
HBOSE	HBOSE SD MP2 Clr*	277	watchtv
HBOSW	HBOSW HD MP4 RTP*	369	watchtv
HBOSW	HBO Sign HD MP4 VQE*	547	watchtv

10 From the SAM Service List window, click **File** and then select **New**. The Set Up SAM Service window opens.

Note: Only the **Short Description** and **Application URL** fields must be configured according to specific rules. For the **Service Name** and **Long Description** fields, enter data that best describes the service.

11 To set up a SAM Service for an application, follow these instructions.

a In the **Short Description** field, type the 4-letter package ID that is used in the config.xml file for the specific application, preceded by "_".

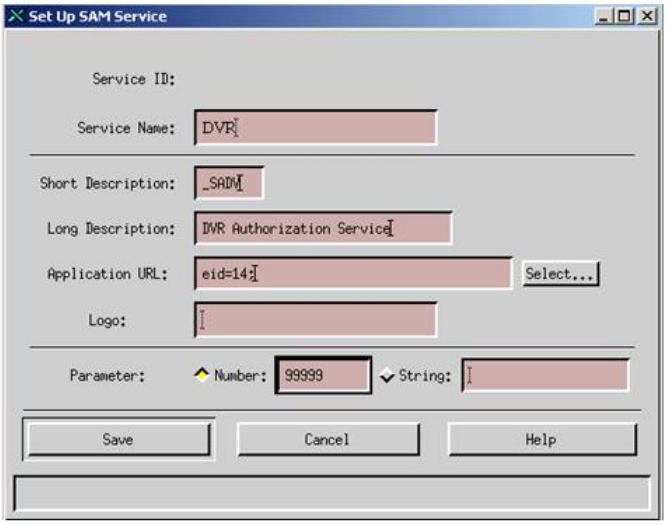
Example: For a package ID of twit, type **_twit** or **_ynws**.

b In the **Application URL** field, type eid=<EID>, using the EID from step 6.

Example: eid=14;

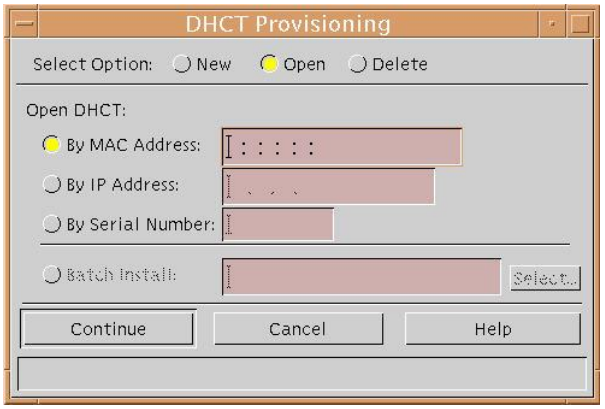
c Click **Save**.

Example: The Set Up SAM Service window should look similar to the following example, which uses the DVR service and has an EID of 14.



Assign Application, Theme, Logo, and Language Packages to STBs

1. From the ISDS tab of the ISDS Administrative Console, select the **Home Element Provisioning** tab.
2. Click **DHCT**. The DHCT Provisioning window opens.



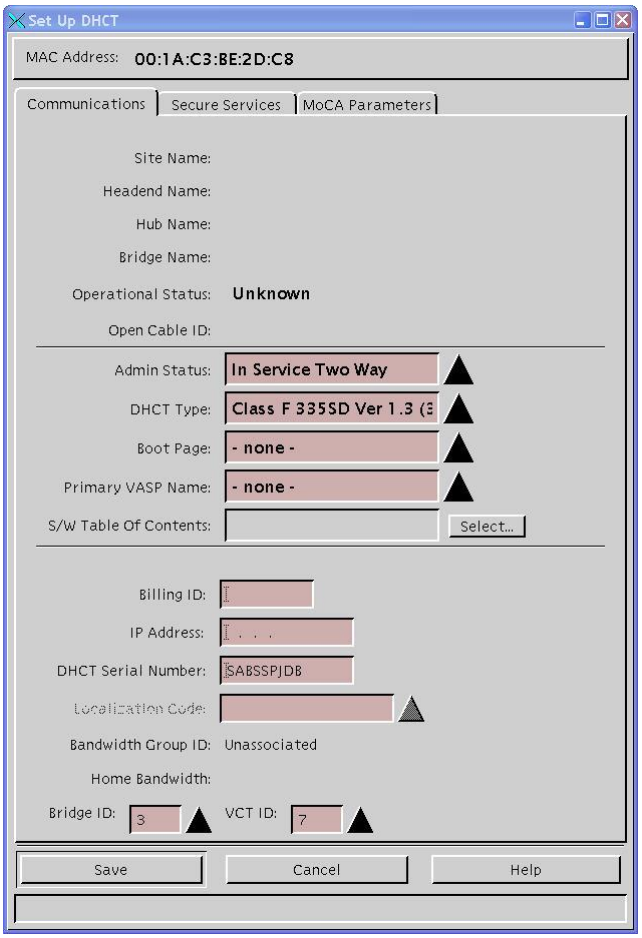
3. Select **New** to create a new DHCT to provision, or select **Open** to provision an existing DHCT.
4. In the **Open DHCT** area of the DHCT Provisioning window, select one of the following options:
   - Click **By MAC Address** to identify the DHCT by MAC address.
   - Click **By IP Address** to identify the DHCT by IP address.

     Notes:

     – The DHCT cannot be identified by IP address if the STB is behind a Network Address Translation (NAT) device.

     – A DHCT cannot be selected by IP address in some networks.
   - Click **By Serial Number** to identify the DHCT by serial number.

5 Click **Continue**. The Set Up DHCT window opens.



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

7 Highlight a package in the **Available** column and then click **Add**. The package moves to the **Selected** column.

8 Click **Save**.

9 Close the Set Up DHCT window.

10 Close the DHCT Provisioning window.

5

Customer Information

If You Have Questions

If you have technical questions, call Cisco Services for assistance. Follow the menu options to speak with a service engineer.

Access your company's extranet site to view or order additional technical publications. For accessing instructions, contact the representative who handles your account. Check your extranet site often as the information is updated frequently.

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

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

July 2012 Printed in USA

Part Number 4036338 Rev B