Consumer Device Software Management for Connected Television

Draft A

28 May 2010

Table of contents

1 Introduction

This document describes how software updates and configuration settings are managed on the device. This document does not cover APIs for access to the configuration information. Mechanisms for remote management and diagnostics are also not within the scope of this specification.

The device is made up of two software components and some corresponding configuration. The Core Device Software is the software image that is installed and managed by the manufacturer. The Platform Software is a software image that contains the top level UI application and can be updated by the platform operator independently of manufacturers and without upgrading the Core Device Software. The device configuration contains specific configuration parameters to be applied to a specific release of Platform Software and can be updated without needing to update either the Core Device Software or the Platform Software.

Figure 1 shows the relationship between these updatable components. Platform Software images can only be installed on a Core Device Software image that implements a specific Platform API Version. Device configuration can only be applied to a specific version of Platform Software.

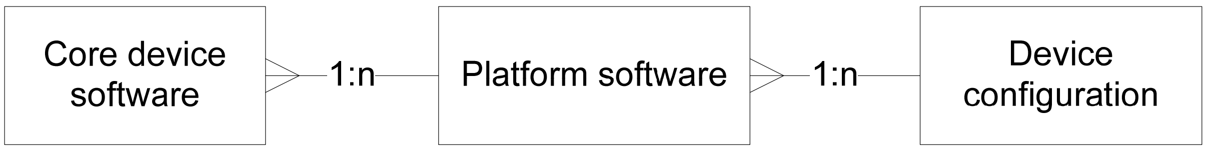

Figure 1. Relationship between software images and configuration

The components that are involved in configuration and software updates are shown in Figure 2 and described in this document. The Local Storage Repository (LSR) is the repository containing the device configuration and is also queried to get information about the state of the device.

For the efficient operation of the platform, a number of configuration parameters are required. A configuration mechanism is specified in section 3 that allows for configuration parameters to be adjusted by a number of configuration sources. The configuration sources are the device manufacturer, the platform operator, the viewer's Internet Service Provider (ISP) and the viewer themselves.

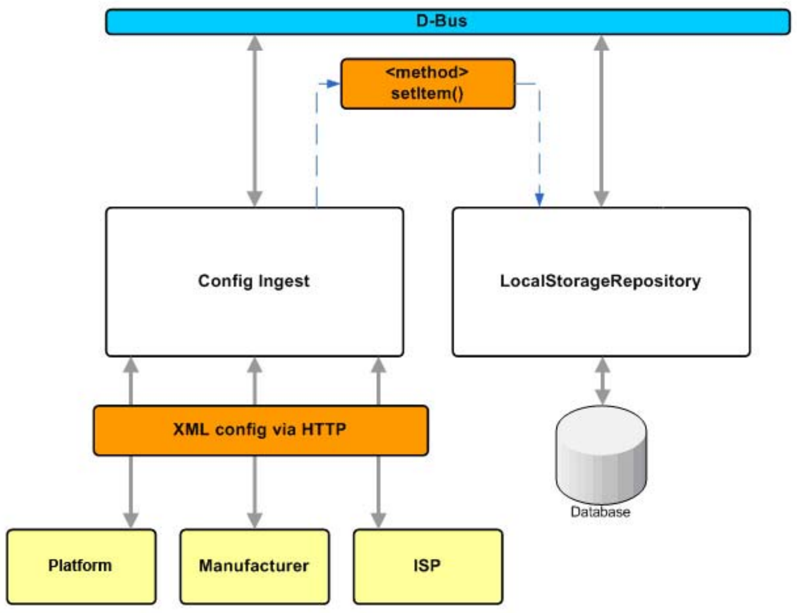

Figure 2. Device architecture showing configuration components

2 Device software upgrade

2.1 Upgradeable modules

The device software is divided into two parts, the Platform Software managed and updated by the platform operator and the Core Device Software managed and updated by the device manufacturer.

The Core Device Software consists of:

- bootloader(s)
- the Linux kernel
- the Linux root filesystem containing libraries and the core device software
- middleware required to support the operation of the device
- presentation engine software including API bindings and data marshalling
- interpreter for platform scripts
- manufacturer-supplied data
- optionally, a default version of the Platform Software
- user interface components and associated assets required for the manufacturer-specific setup and configuration of the device
- manufacturer's default configuration parameters. See section 3.2.1
- public key certificates

The Platform Software consists of:

- default platform configuration
- the top level UI application (including all images and data needed for off-line use)
- platform scripts providing soft logic for device components providing background functions and plug-in functionality for system services
- static data files supporting these elements

The Platform Software is architecture independent and contains no native compiled executable code.

The device shall include a default version of the Platform Software so that when the device starts for the first time the top level UI is available.

The device may include a default version of the Platform Software within the Core Device Software.

If the Core Device Software does not include a default version of the Platform Software the device shall include the default Platform Software separately.

This is illustrated in the Figure 3:

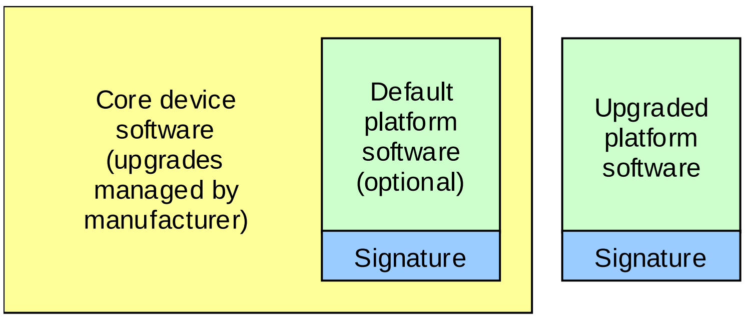
Core device software (upgrades managed by manufacturer)
Default platform software (optional)
Signature
Upgraded platform software
Signature

Figure 3. Device software images

2.2 Core Device Software upgrade

Devices shall support the upgrade of the Core Device Software over broadcast using the DVB-SSU Firmware Update Method described in D-Book 6.2.1 Section 23.6.

Devices shall support upgrading of the software over the IP connection. Manufacturers are responsible for implementing a mechanism for secure and reliable upgrade.

Software images shall be encrypted in delivery over broadcast and IP and on any servers on which it is hosted.

After a successful installation of the Core Device Software the Local Storage Repository (LSR) shall contain:

- the manufacturer's name
- the manufacturer's OUI
- model number
- Platform API Version number
- Core Device Software version number

The manufacturer's software may be divided into separate parts for the purposes of upgrading.

After upgrading the Core Device Software the device shall check for a newer version of the Platform Software as defined in section 2.3.

2.3 Platform Software update

Devices shall support the updating of Platform Software using both the broadcast and broadband update mechanisms described in this section.

Devices shall download a new Platform Software image using either the

IP Platform Software download method described in section 2.3.1 or the DVB-SSU Platform Software download method described in section 2.3.2.

When a new Platform Software image has been successfully downloaded and is available locally, the device shall save a local copy of the new Platform Software image.

The Platform Software image shall be in the format described in section 2.3.3.

Once a new Platform Software image is available locally the device shall verify that the image is signed by the platform operator, decrypt and install it using the process described in section 2.3.4.

Devices shall store at least two versions of the Platform Software – one default version and at least one downloaded from the update server. Devices shall use the updated copy in preference, provided that it has been verified and successfully installed.

The device shall only install Platform Software if the Core Device Software implements the Platform API Version specified in the header of the Platform Software.

The Platform Software is provided by the platform operator as a single signed and encrypted image. Devices may need to consider splitting the software after downloading if other requirements dictate that some parts are held in flash memory and others on hard disk storage.

The viewer shall not be notified or asked to confirm a software and configuration updates. However, the device setup may include an option through which the viewer can explicitly disable automatic software updates.

The device shall not inform the viewer when a software update or configuration update fails.

2.3.1 IP Platform Software download method

The device may be configured to only perform configuration and software update checks during a pre-defined download window.

The device shall check for a new Platform Software image at the regular update interval configured in the Local Storage Repository.

If the device has not successfully checked for new Platform Software for seven or more days the device shall ignore the download window and start a check now.

If the download window is defined then the device shall schedule a Platform Software update check at a random time within the daily download window.

The device shall check for a new Platform Software image and download it using the following method.

> The device shall check for a new platform image using a HTTP/1.1 GET request with an If-Modified-Since header quoting the modification date and time of the currently installed Platform Software image. If an Etag header was provided with a previous response, the device shall also use an If-None-Match header quoting the Etag.
>
> The device shall check for a new Platform Software image and download it using the following constructed URI:
>
> > http://[baseuri]/[manufacturer]/[model]/[platform_api_ver]/[image_name].bin
> >
> > Where the values are retrieved from the Local Storage Repository using these names
> >
> > > baseuri = platform.software.update.baseuri
> > >
> > > manufacturer = hardware.manufacturer
> > >
> > > model = hardware.model
> > >
> > > platform_api_ver = core.software.platformapi.version
> > >
> > > image_name = platform.software.update.imagename
> > >
> > > Note: These are proposed parameter names and are subject to change.
>
> If a new Platform Software image is available the response shall be a HTTP/1.1 200 OK response with a Content-Type of application/octet-stream. The body of the HTTP response shall be the Platform Software image in the format described in section 2.3.3. The device shall then save a local copy of the new Platform Software image and update the platform.software.update.lastchecked item in the LSR.
>
> If a new Platform Software image is not available the response shall be a HTTP/1.1 304 Not Modified. The device shall then update the platform.software.update.lastchecked in the LSR.

If a new Platform Software image is downloaded the device shall also download the Platform Software digital envelope (defined in section 2.3.3.1) using the following URI:

> http://[baseuri]/[manufacturer]/[model]/[platform_api_ver]/[platform_software_ver].env
>
> Where: platform_software_ver is the Platform Software version number extracted from the Platform Software Image Descriptor (PSID) of the downloaded Platform Software. The PSID is defined in section 0.

The base URIs of the software update service and Platform Software update interval can be changed by the configuration. However, in the event that a software update service responds with an error on five consecutive occasions, the device shall revert to the previous location for the next attempt.

Platform Software downloads shall occur at a randomly selected time within the download window. Standard back-off mechanisms for HTTP requests shall be implemented by the device in the event of download errors as defined in the IP Content Delivery 1108-S specification.

Once the device has successfully downloaded a Platform Software image it shall install the image using the procedure described in section 2.3.4.

2.3.2 DVB-SSU Platform Software download method

Devices shall support the upgrade of the Platform Software using the DVB-SSU Firmware Update Method described in D-Book 6.2.1 Section 23.6 except for the following differences:

> The DSI in the data carousel will contain the platform operator's OUI rather than the manufacturers OUI.

Given the expected time for a device to complete the download, the device must be able to perform the download as a background operation while the user is watching TV or making a recording. If the viewer's actions force all available tuners to retune to multiplexes which are not carrying the download then the download will be interrupted. The device shall be able to resume partially completed downloads rather than having to re-start from the beginning.

2.3.3 Platform Software packaging format

The Platform Software image shall be a signed, encrypted *cramfs* filesystem image.

The Platform Software packaging format describes the format of the files used to upgrade the Platform Software.

The Platform Software update package contains a Platform Software digital envelope and a Platform Software image.

2.3.3.1 Platform Software digital envelope

The Platform Software digital envelope shall be of type CMS Enveloped-Data with detached content as described in RFC 5652 Section 6.

2.3.3.2 Platform Software image

The Platform Software image shall be of type CMS Signed-data as described in RFC 5652 Section 5, with a pre-pended Platform Software Image Descriptor (PSID). It will be encrypted as described in section 2.3.3.6.

The Platform Software image shall be of the form:

Item	**Offset (bytes)**	**Size (bytes)**
Platform Software Image Descriptor (PSID)	0x00	0x20
{ CMS Signed-data }$^{\text{CEK (AES-256-CBC)}}$ (as per: section 2.3.3.6 and RFC 5652 Section 5)	0x20	variable

2.3.3.3 Platform Software Image Descriptor (PSID)

The Platform Software Image Descriptor (PSID) shall be of the form:

Item	**Contents**	**Size (bytes)**
Magic cookie	0xCA, 0xA5	0x02
Platform Software packaging format version identifier	0x00, 0x00, 0x00, 0x01	0x04
Unused	< unused >	0x0A
Required Platform API major version	API major version (e.g. 0x02)	0x01
Required Platform API minor version	API minor version (e.g. 0x01)	0x01
Required Platform API revision	API revision (e.g. 0x00, 0x00)	0x02
Platform Software image major version	Platform Software major version (e.g. 0x02)	0x01
Platform Software image minor version	Platform Software minor version (e.g. 0x01)	0x01
Platform Software image revision	Platform Software revision (e.g. 0x00, 0x00)	0x02
Capabilities	0x00 (all 8 bytes)	0x08

The Platform Software packaging format version identifier is used to indicate the version of the packaging format used by the Platform Software image.

Versions of the Platform Software shall be identified by major, minor and revision numbers and define the Platform API version that it is required.

The device shall only install Platform Software that matches the Platform API version of the Core Device Software.

The Platform Software image version number will increment with each new release.

The Capabilities item is a reserved section of the PSID which will be used at a later date to indicate the device capabilities which are compatible with a specific version a Platform Software image.

2.3.3.4 Encapsulated Content

The eContent section (RFC5652 Section 5.2) of the CMS Signed-data section shall be of the form:

Item	**Offset (bytes)**	**Size (bytes)**
Platform Software Image Descriptor (PSID)	0x00	0x20
CramFS image	0x20	Variable

The PSID is duplicated in the encapsulated content section since this part of the message is verifiable.

2.3.3.5 Signing

The platform operator shall provide a Platform Software signing X.509 (RFC 5280) certificate containing an RSA 2048 bit public key in the SignerInfo section of Platform Software image.

The content section of the Platform Software image shall be signed using the private key paired with this public key.

2.3.3.6 Encryption

Manufacturers shall provide the platform operator with either a single Key Encryption Key (KEK), or a number of KEKs. Each KEK shall be a 2048 bit RSA public key.

The platform operator shall generate a new Content Encryption Key (CEK) for each distribution of the Platform Software image. The CEK shall be a 256 bit key.

The CEK shall be encrypted by the platform operator once for each KEK provided to the platform operator. Each encrypted key (i.e. $\{CEK\}^{KEK}$) shall be stored in a KeyTransRecipientInfo structure of the Platform Software digital envelope.

The Platform Software image will be encrypted using the CEK. The encryption algorithm used shall be AES-256-CBC.

2.3.4 Platform Software installation process

The Platform Software installation process is as follows:

1. Download the new Platform Software image and Platform Software digital envelope as described above.
2. Check the preconditions on the plaintext PSID in the Platform Software image (as described in section 2.3.4.1)
3. If preconditions are met then decrypt (as described in section 2.3.4.2) and verify using the signature provided (as described in section 2.3.4.3) the Platform Software image
4. Check the preconditions on the verified PSID (as described in section 2.3.4.1)
5. If the preconditions are met on the verified PSID then;
    a. Copy the Platform Software image to a place it can be mounted from.
    b. Take a copy of the signature and certificate chain to use when verifying the Platform Software image.
    c. The next time the device leaves standby mode it shall switch to the latest locally installed version of the Platform Software image.
    d. Under normal operation the device successfully verifies the Platform Software image using the signature provided before mounting it as a cramfs[1] filesystem.

The Platform Software shall not contain any native executable code. For added security, it is therefore recommended that the filesystem image be mounted without execute permission.

Once the image has been verified, it can be mounted into the Linux filesystem of the device, either directly, or using the loopback driver as appropriate.

If the viewer has disabled automatic standby on idle, the device will still alert the viewer that the device will enter standby and the software update will take place when no recording or other activity is taking place. The user may be presented with an option to defer the standby/upgrade procedure but not to cancel it.

If the signature verification fails, the device shall fall back to the last known good version of the Platform Software.

[1] cramfs is a Linux filesystem designed to be simple, small, and to compress things well. It is used on a number of embedded systems and small devices. (http://sourceforge.net/projects/cramfs/)

2.3.4.1 Preconditions

The device shall assert the following preconditions on the PSID and the verified PSID before installing a Platform Software image and making it available for use:

1. Assert the magic cookie is 0xCA, 0xA5
2. Assert the Platform Software packaging format version identifier is 0x00, 0x00, 0x00, 0x01
3. Assert the required Platform API version is equal to the version of the Platform API implemented by the Core Device Software
4. Assert the Platform Software image version is greater than the installed Platform Software image version
5. Assert all 8 bytes of the capabilities section are 0x00

2.3.4.2 Decryption

The Content Encryption Key (CEK) shall be decrypted using $KEK_{private}$, i.e. the private key paired with KEK.

The Platform Software image shall be decrypted using the CEK. The algorithm used shall be AES-256-CBC.

$KEK_{private}$ shall be stored as a device secret.

2.3.4.3 Verification

Devices shall calculate a message digest of the signed content and verify the digital signature as per RFC 5652 Section 5.6.

Devices shall assert that the Signed-data section (RFC 5652 Section 5) has been signed by the platform operator according to the overarching trust model.

2.3.4.4 Certificate revocation

Manufacturers shall assert that no certificate in the signer's certificate chain has been revoked using the certificate revocation list (CRL) installed in the Core Device Software.

3 Device configuration

3.1 Introduction

For the efficient operation of the platform, a number of configuration parameters are required. A configuration mechanism is specified that allows for configuration parameters to be adjusted by a number of configuration sources. The configuration sources are the device manufacturer, the platform operator, the viewer's Internet Service Provider (ISP) and the viewer themselves. To enable this, the device configuration contains a set of defined URI prefixes from which configuration data must periodically be gathered. To these prefixes, the device appends information about its manufacturer, model, current configuration version and current Platform Software version to arrive at a complete URI to use for the configuration request. Configuration data retrieved from these URIs will be signed.

A set of named configuration parameters and the configuration sources that have permission to set each of these parameters is defined by the platform operator.

The various sources of configuration data are processed in a specific order to arrive at a merged database of configuration values. The parameters that a configuration source can set are listed in the configuration signing certificate issued to the configuration source. The order the configuration sources are applied in and the parameters listed in each signing certificate determine which configuration sources are able to set and overwrite specific parameters. When manufacturer and ISP configuration files are processed, the valid parameters listed in the signing certificate are used to ensure manufacturers and ISPs only set or overwrite parameters they have permission for. This merging operation is repeated when any one of the configuration sources has changed.

3.2 Acquisition of configuration information

Configuration information is obtained from a number of configuration sources and applied in a specific order. The order that configuration sources are applied and the set of configuration permissions are used to control which of the configuration sources is able to set specific groups of parameters.

3.2.1 Local configuration sources

The device shall contain two local sources of configuration, the manufacturer's default configuration and the platform operator's default configuration.

The manufacturer shall apply its default configuration to the Local Storage Repository before applying any other local or remote configuration.

The platform operator's default configuration shall be contained in the Platform Provisioning File provided as part of the Platform Software.

3.2.2 Remote configuration sources

The remote configuration sources used to configure the device as follows and are applied in the order defined here.

1. Platform Configuration File is provided on the network by the platform operator at a pre-defined location. This location defines the URI the device uses to check for updates to the platform configuration and download the new configuration file if one exists.
2. Manufacturer Configuration File is provided on the network by the manufacturer at a pre-defined location. This location defines the URI the device uses to check for updates to the manufacturer configuration and download the new configuration file if one exists.
3. ISP Configuration File can be optionally provided on the network by the viewer's ISP if the ISP is a participating ISP. The device uses a single pre-defined URI to check for updates to the ISP configuration and download the new configuration file if one exists. Redirection or other means allow the file to be served by the ISP themselves.

The following diagram illustrates how the various sources of configuration information are managed:

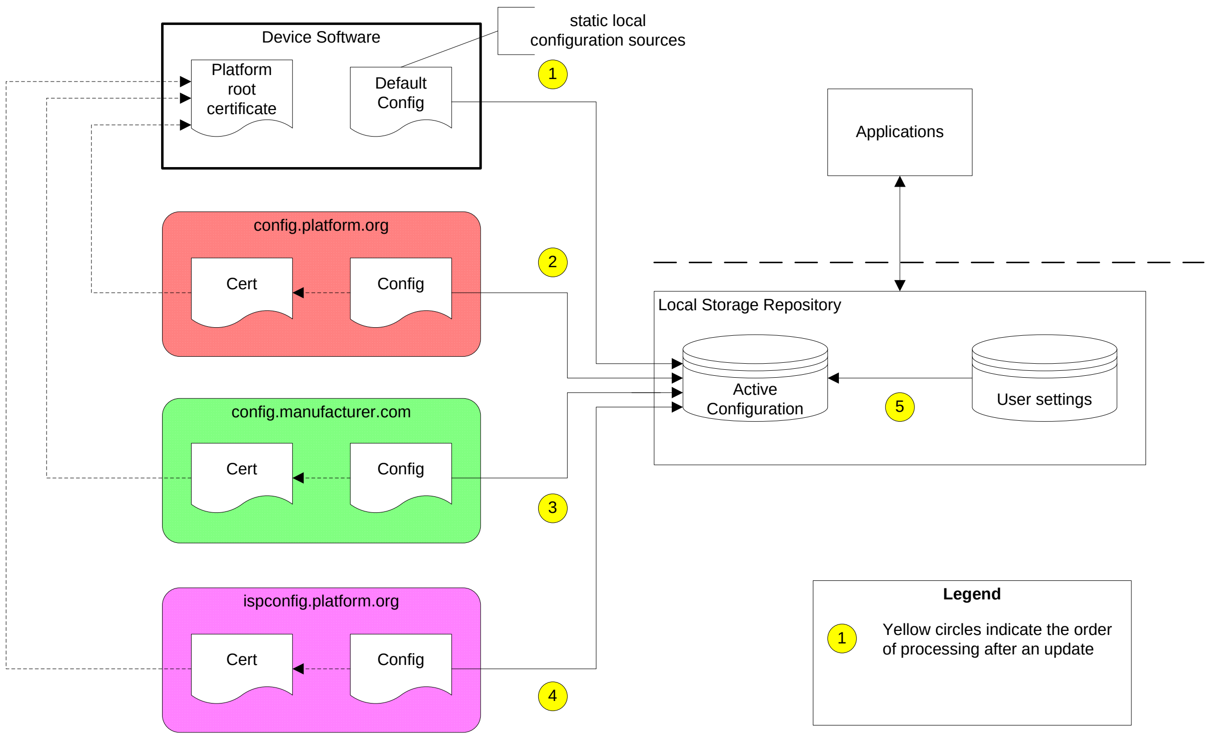

Figure 4 Configuration sources and their processing order

3.2.3 File format

The configuration sources listed in section 3.2.1 shall use the same file format.

Configuration files are provided as a simple XML document. Annex A defines the schema and includes an example configuration file.

The configuration file is signed using the method described in section 3.2.4

3.2.4 Authentication

The platform operator manages a configuration signing CA certificate.

Each configuration file downloaded to the device shall be signed by the organisation that created it using their configuration signing key and packaged with their corresponding certificate signed by the platform operator.

Devices shall successfully verify the signature before applying the configuration parameters. If verification of the signature fails or verification of the certificate chain fails, as defined in the overarching trust model, the configuration parameters shall not be applied to the device.

The signature method is CMS as defined in RFC 5652 with an unencrypted data section and using RSA.

The configuration signing certificate shall be identified as the SignerIdentifier, as described in RFC5652 section 5.3.

This configuration signing CA certificate shall be provided in the CMS Certificates section.

3.2.5 Certificate revocation

Manufacturers shall assert that no certificate in the configuration signer's certificate chain has been revoked using the certificate revocation list (CRL) installed in the Core Device Software.

3.2.6 Access control

The list of parameters that a configuration source is allowed to set is defined in an X.509v3 Arbitrary Extension in each configuration signing certificate. The extension allows a set of parameter names and wildcards to be specified. e.g. net.delivery.* to allow an ISP to set any parameters matching that name.

The X.509v3 extension shall be named "configurationPermissions" and have an Object Identifier (OID) provided by the platform operator.

The "configurationPermissions" extension shall be of ASN.1 type SET OF < UTF8String >.

3.2.7 Download

Devices shall download new configuration data at regular intervals by using an HTTP/1.1 GET request with an If-Modified-Since header quoting the date and time at which the last successful download of the relevant configuration file was achieved. If an Etag header was provided with a previous response, the device shall also use an If-None-Match header quoting the Etag.

Devices shall follow HTTP redirections.

The device shall check for a new configuration update and if necessary apply the changes each day as specified by the update interval defined in the local storage repository. The default setting for this is provided in the platform provisioning file. Configuration updates can alter the configuration update interval for the platform operator, manufacturer and ISP. Configuration update requests shall only occur at a randomly selected time within the configuration and update download window. Standard back-off mechanisms shall be used in the event of download errors. Refer to the IP Content Delivery specification for more details.

If a device has not successfully checked for a new configuration update for seven or more days it shall perform a configuration update.

Configuration updates can change the URI of the configuration update service. If a configuration update service URI fails five times the device shall use URI provided as part of the default platform configuration.

3.2.8 Location

Devices shall not proceed with a configuration update until it has successfully downloaded a Platform Configuration File.

The device appends information about its manufacturer, model, current configuration version and current Platform Software version to the base URI to arrive at a complete URI to use for the configuration request.

> http://[baseuri]/[manufacturer]/[model]/[platform_software_ver]/[config_name].xml
>
> Where:
>
> > [platform_software_ver] is the Platform Software version number extracted from the Platform Software Image Descriptor (PSID) of the downloaded Platform Software. The PSID is defined in section 0.
> >
> > [config_name] is the name of the configuration to be applied retrieved from the Local Storage Repository.

The base URI of the configuration files can be changed by the configuration. However, in the event that a configuration file cannot be obtained from the modified location on five consecutive occasions, the device shall revert to the default base URI for the next attempt.

The device shall be configured with a secondary base URI for configuration updates and if configuration updates fail using the first base URI the device shall attempt the update using the secondary base URI.

Devices shall process manufacturer or ISP configuration files in the following way:

- If the download fails due to an authoritative response that the domain name (or a domain name referenced in an HTTP redirect) does not exist, the device shall treat that source as providing an empty configuration file and continue with the configuration update
- If the download fails for any other reason (including but not limited to any non-authoritative DNS error, TCP/IP connection error, HTTP 4xx or 5xx response), devices shall treat this as a transient failure and continue to use the most recent successfully-merged configuration file for that configuration source, if one is available.

The URI to be used for the ISP configuration update is determined using a separate ISP discovery mechanism.

3.2.9 Configuration update process

When the device determines that a configuration source has changed and has downloaded the new configuration using the method in section 3.2.7, the device shall update the configuration database using a process equivalent to that described below:

1. Start a transaction so that configuration updates can be applied atomically.
2. Initialise the configuration database with the default configuration taken from the platform provisioning file that forms part of the Core Device Software
3. For each of the last downloaded configuration files for platform, manufacturer and ISP configuration in that order:
    a. Verify that the signature of the configuration file is valid, and the certificate is issued by the configuration certificate and has not been revoked. If successful, for each parameter:
        i. Check that the parameter name matches an element in the configurationPermissions X.509 certificate extension.
        ii. If the source has the permission to set this parameter or the configurationPermissions extension doesn't exist and the parameter has changed then update the configuration value in the database with the value from the configuration file.
4. If all configuration updates were successful then commit the configuration update transaction, otherwise return the configuration database to its previous state.
5. Activate the newly updated configuration database.
6. For all parameters that have changed create a notification of the change.

3.3 Configuration parameters

The set of valid configuration parameters are defined in Annex B.

Annex A Configuration XML file format

Annex A.1 Configuration XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://refdata.myplatform/schemas/myplatform-config/2010-05-05"
    targetNamespace="http://refdata.myplatform/schemas/myplatform-config/2010-05-05">

    <xsd:element name="config" type="Config"></xsd:element>
    <xsd:complexType name="Config">
        <xsd:sequence>
            <xsd:element name="item" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:simpleContent>
                        <xsd:extension base="xsd:string">
                            <xsd:attribute name="name" type="xsd:string"></xsd:attribute>
                        </xsd:extension>
                    </xsd:simpleContent>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="version" type="xsd:int"></xsd:attribute>
    </xsd:complexType>
</xsd:schema>
```

Annex A.2 Platform provisioning file example

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cfg="http://refdata.myplatform/schemas/config/2010-05-05"
        xsi:schemaLocation="http://refdata.myplatform/schemas/config/2010-05-05 myplatform-
config.xsd"
        version="1">
    <item name="platform.software.name">Platform Software v2.1</item>
    <item name="platform.software.version">1</item>
   <item name="platform.software.update.baseuri">http://myplatform/config/</item>
    <item name="platform.config.update.interval">3600</item>
    <item name="platform.software.autoupdates"></item>
</config>
```

Annex B Configuration parameters

This list will be extended with additional configuration parameters in a subsequent revision of this specification.

Configuration Item	Type	Description
platform.		Namespace for platform configuration
platform.software.name	string	The name and version of the current Platform Software
platform.software.version	32bit int	The version number of the current Platform Software
platform.software.etag	string	The Etag associated with the current Platform Software image when it was downloaded
platform.software.update.baseuri	uri	The Platform Software update base URI. Used to create the HTTP request required to check for a newer version of the Platform Software.
platform.software.update.interval	unsignedInt	The time in seconds that the device waits before checking for a new version of the Platform Software
platform.software.update.lastupdated	date	The date and time that the current Platform Software was successfully updated
platform.software.update.lastchecked	date	The date and time that the device last checked for the availability of new Platform Software
platform.software.update.window.start	time	The start time of the Platform Software update window. If configured the Platform Software and configuration checking and downloads shall be scheduled to start at a random time within the download window.
platform.software.update.window.end	time	The end time of the Platform Software update window. If configured the Platform Software and configuration checks and downloads shall be scheduled to start at a random time within the download window.
platform.software.autoupdates	boolean	Enables automatic software and configuration updates
platform.config.name	string	The name and version of the current platform configuration
platform.config.version	unsignedInt	The version of the current platform configuration
platform.config.etag	string	The Etag associated with the current platform configuration file when it was downloaded
platform.config.baseuri	uri	The URI used to check for and download the latest platform configuration
platform.config.updateinterval	unsignedInt	Time in seconds the device waits before checking for a new platform configuration
platform.config.updated	date	The date and time that the platform configuration was successfully updated
platform.config.lastchecked	date	The date and time that the platform last checked for an updated configuration

Configuration Item	Type	Description
core.software.name	string	The name of the Core Device Software currently installed
core.software.version	32bit int	The version of the Core Device Software currently installed
core.software.platformapi.version	32bit int	The Platform API version implemented by the current Core Device Software. Platform Software determine if they are compatible with the installed Core Device Software by checking this version number.
core.software.lastupdated	date	The date and time that the Core Device Software was installed
manufacturer.name	string	The name of the device manufacturer
manufacturer.oui	OUI	The OUI of the device manufacturer
manufacturer.config.name	string	The name of the current manufacturer configuration release
manufacturer.config.version	unsignedInt	The version of the current manufacturer configuration
manufacturer.config.etag	string	The Etag associated with the current manufacturer configuration file when it was downloaded
manufacturer.config.location	uri	The URI used to check for and download the latest manufacturer configuration
manufacturer.config.updateinterval	unsignedInt	The time in seconds the device waits before checking for a new manufacturer configuration
manufacturer.config.updated	date	The date and time that the manufacturers configuration was last sucessfully updated
manufacturer.config.lastchecked	date	The date and time that the device last checked for a new manufacturers configuration
hardware.model	string	The device model number