netReady™ Factory Process Guide

Factory Process Considerations for your netReady Equipped TV

Version 2.0.4

UpdateLogic Incorporated
508-624-8688 (TEL)
508-624-8686 (FAX)
http://www.updatelogic.com

Thank you for using the UpdateTV netReady™ software update solution. UpdateLogic Incorporated provides a comprehensive, cost-effective mechanism, which can update the complex firmware within a digital television by sending firmware updates over the Internet, digital broadcast infrastructure, and via USB.

If you find any problems with this software, please report them via email to techsupport@updatelogic.com.

Please visit our website at www.updatelogic.com for the latest news and information.

UpdateLogic Incorporated

508-624-8688 (TEL)

508-624-8686 (FAX)

contact@updatelogic.com

Table of Contents

1 NetReady Factory Process Guide

This document is designed to introduce UpdateLogic's netReady technology to your factory staff prior to production so that they can begin to become familiar with connected TV production issues. It provides a system overview, describes the NOC interface, and outlines production process guidelines that are designed to help factory staff test and troubleshoot netReady-related connected TV issues.

1.1 Change Tracking

Version Number of This Document	**Changes**
2.0.4	New OOB NOC Device Group description.
2.0.3	Improved the robustness of the sample code.
2.0.2	Change the steps order in "3.1.4 Creating the OOB Update"
2.0.1	Factory process checklist. Example ULPK CSV extraction code.
2.0.0	Original

1.2 Scope

This document assumes that the reader is *not* familiar with netReady technology, but is well versed in digital television technology including digital receiver architecture, Internet and MPEG protocols, and general connected TV concepts. This document does *not* describe creating or managing updates or any issues related to porting the Agent or using the UpdateTV tools.

1.3 Audience

This document is designed to be read by a CEM's factory staff and their management

1.4 Related Documents

All netReady documents are distributed with the netReady SDK. The netReady SDK is distributed with documentation that appeals to a wide audience. Only the most technical staff members need to read all the documentation.

Document	Topic	Source
netProvision Technical Guide	Description of netProvision function and connected device integration. Useful to all audiences.	ULI
netReady Intro	General introduction to the UpdateTV technology. Useful to all audiences.	ULI
netReady Agent Integration Requirements	Target software and hardware requirements for your netReady device.	ULI
netReady Agent Porting Instructions	Step by step porting instructions for the Agent. Useful to receiver Agent porting engineers.	ULI
netReady Agent Validation Guide	Description of tests the CEM should run to validate their port of the netReady Agent to a CTV. Useful to all audiences.	ULI
netReady Factory Process Guide	Introduces netReady technology to your factory staff prior to production	ULI
netReady Technical Support Guide	Introduces netReady technology to your technical support staff prior to production	ULI
netReady NOC User's Guide	User's manual for interacting with the Network Operating Center via its web interface. Useful to carousel development and test engineers.	ULI
netReady Publisher	User's manual for the Publisher utility. Useful to all audiences.	ULI
netReady Technical Support Guide	An introduction to netReady and supporting CTVs for a CEM's technical support organization.	ULI
netReady Factory Process Guide	An introduction to netReady and manufacturing CTVs for a CEM's factory staff.	ULI
UpdateTV TestCarouselBuilder	User's manual for the TestCarouselBuilder utility. Useful to all audiences.	ULI
UpdateTV TSFactory	User's manual for the TSFactory utility. Useful to technology evaluators and engineers supporting factory updates.	ULI
UpdateTV StreamViewer	User's manual for the StreamViewer utility. Useful to all audiences.	ULI
UpdateTV Factory Update Guide	A document that explains the use of factory mode from an operational and programmatic point of view. Useful to technology evaluators and engineers supporting factory updates.	ULI
UpdateTV A/97 Implementation Specification	UpdateTV implementation of A/97 download data service. Useful to technology evaluators and advanced users.	ULI
ATSC A/97: Software Download Data Service http://www.atsc.org/standards.html	ATSC spec. Useful to technology evaluators and advanced users.	ATSC

ATSC A/90: Data Broadcast Standard http://www.atsc.org/standards.html	ATSC spec. Useful to technology evaluators and advanced users.	ATSC

1.5 Glossary of Terms

Term	Description
API	Application Programming Interface.
ASI	Asynchronous Serial Interface, often used to distribute MPEG transport streams.
ATSC	Advanced Television Standards Committee. www.atsc.org
Back End	MPEG decode, graphics and user interaction portion of the digital receiver hardware.
BEM	Broadcast Encoder Monitor. A 1U rack mount device that accepts ASI or SMPTE DTV MPEG data in and inserts the UpdateLogic broadcast carousel. Also contains an RF input to monitor the resulting output. This device is installed at both broadcast sites and in customer labs.
CableLabs	Research and development consortium that is dedicated to helping its cable operator members integrate new cable telecommunications technologies. www.cablelabs.com
CAB File	An archive file like a ZIP file that contains one or more files in compressed format.
CEM	Consumer Electronics Manufacturer
CHM	Customer's Hardware Model
Component	A separately updateable element within an update.
CMG	Customer's Model Group
Content Service Provider	Content Service Provider. For instance, Netflix, Vudu, Amazon, CinemaNow, etc.
COUI	Customer's Organizational Unique Identifier
CSP	Content Service Provider.
CTV	Connected TV. A TV that supports an Ethernet and/or WiFi connection to the internet and uses that connection to connect to services like streaming media, news, weather, etc..
DCR	Digital Cable Ready. This acronym refers to consumer electronic devices equipped with Digital Tuners and Cable Card slots for use in receiving one-way Digital Cable signals and features.
Front End	Tuner and demodulation portion of the digital receiver hardware.
Hardware Model	A number describing a sub-set of a model group that is usually panel size specific.
HM	Hardware Model
HMR	Hardware Model Range
Image	An entire software or data entity that is broken into pieces

	called “modules” when broadcast on the network.
Manufacturers’ ESN	A serial number that is readable by software and also printed on a label affixed to the outside of a TV. Used by the consumer to identify their TV to technical support staff. Each CEM has a different serial number scheme. The customer can display the serial number by using the TV’s menus. Also called “the SN”.
Manufacturer’s Software Version	A string that contains version information like “TV_System_3.7”. The customer can display the software version using the TV’s menus.
Model Group	A number that uniquely defines a particular TV model within a give OUI (company). A model group may be sub-divided into different hardware models.
MG	Model Group
Module	Name of the distribution unit that a software or data “image” is broken down into in order to broadcast it efficiently on the network.
MV	Module Version
netProvision	A facility that is part of netReady used for delivering streaming media security credentials to a TV.
netReady	A suite of connected TV services that includes netUpdate, netProvision, netRegister, and netDiag.
netReady Agent	Software embedded in a digital TV receiver to enable the reception of software updates via the Internet, Broadcast, and USB.
netUpdate	A facility that is part of netReady used for delivering security credentials to a device in the consumer’s home.
NOC	The UpdateTV Network Operations Center. A server farm that distributes updates over the Internet and broadcast networks.
NOC Web Interface	A web-based application that is used to manage updates and provisioning over the UpdateTV Network. The URL for this application is extdev.updatelogic.com for the development NOC and support.updatelogic.com for the production NOC. Often called simply “the NOC”.
OAD	Over Air Download (update via terrestrial RF broadcast).
OOB Testing	Out Of Box Testing
Open Cable	Standards organization setting standards for Open Cable networks and related technology. http://www.opencable.com
OTA	Over The Air (i.e. Terrestrial Broadcast).
OUI	Organizational Unique Identifier
PSI	Program Specific Information. Generally refers to the PAT, PMT, CAT, NIT, and TDT tables in an MPEG stream.
PSIP	Program and System Information Protocol. Generally refers to the STT, MGT, VCT, RRT, EIT, ETT, and DCCT

	tables in an MPEG stream.
Publisher	A command line tool provide by UpdateLogic for the purpose of packaging a set of modules for software distribution.
Registration	The process that all TVs go through when they first connect to the Internet after shipment in order to register their identity with the UpdateLogic NOC and get a network identity. During this process the device's serial number is sent to the NOC and it receives a ULID that is used for all subsequent conversations with the NOC.
Re-registration	A process that can be initiated via the NOC web interface by support staff in order to force a TV to re-register and receive its security credentials again.
Registration	The process that all TVs go through when they first connect to the Internet after shipment in order to register their identity with the UpdateLogic NOC and get a network identity. During this process the device's serial number is sent to the NOC and it receives a ULID that is used for all subsequent conversations with the NOC.
RTOS	Real Time Operating System.
Security Credential	A TV-unique or model group-common file that a TV must be provisioned with in order to use the associated streaming service that relies on it.
Streaming Service	An application that provides streaming video or audio.
StreamViewer	A software tool supplied by UpdateLogic to view and analyze file based transport streams that contain compatible carousels.
SV	Software Version
SVR	Software Version Range
Update	An entity containing multiple components that may be software or data that replaces the components that were shipped with the TV. Used to fix bugs and add features.
UpdateTV Agent	Software embedded in a digital TV receiver to enable the reception of software updates via the UpdateTV Network.
UpdateTV Network	A collection of networked data insertion servers, and related broadcast equipment, used to distribute software updates to UpdateTV enabled receivers. The network provides for distribution via over-the-air and Open Cable digital networks environments.
UpdateTV Receiver	A digital television receiver enabled with the UpdateTV Agent so as to receive software updates from the UpdateTV Network.
UpdateTV Server	A computing device used to generate a data stream that can be multiplexed with other digital program content at a broadcast facility for inclusion in the ATSC broadcast stream.
TestCarouselBuilder (TCB)	A Win32 GUI based tool that creates test carousel transport

	streams from template test files and Publisher packaged software updates.
Unit Under Test (UUT)	The CEM's target receiver that is being tested.

2 Factory Production of Connected Devices

Connected devices present a manufacturing challenge. Although factories have been creating modern HD TVs , Blueray players, etc. for some time now, connected devices are inherently more complex. This complexity comes from the following factors:

- Connected devices depend on an Internet connection which may be unreliable or have limiting characteristics outside of the device manufacturer's control.
- Even when the Internet connection is dependable connected devices rely on 3rd party services that may cause problems on the connected device when they're not running correctly.
- Connected devices have security requirements that lock them down and can make running diagnostics difficult.
- Connected devices take longer to power up and down which can force changes in production line logistics.
- Connected devices are generally a new technology with emerging and sometimes surprising behavior.
- Connected device out of box (OOB) testing is more complex simply because there are more features to test.

This document recommends guidelines for factory manufacturing and testing processes that address the issues above issues. It focuses primarily on OOB testing.

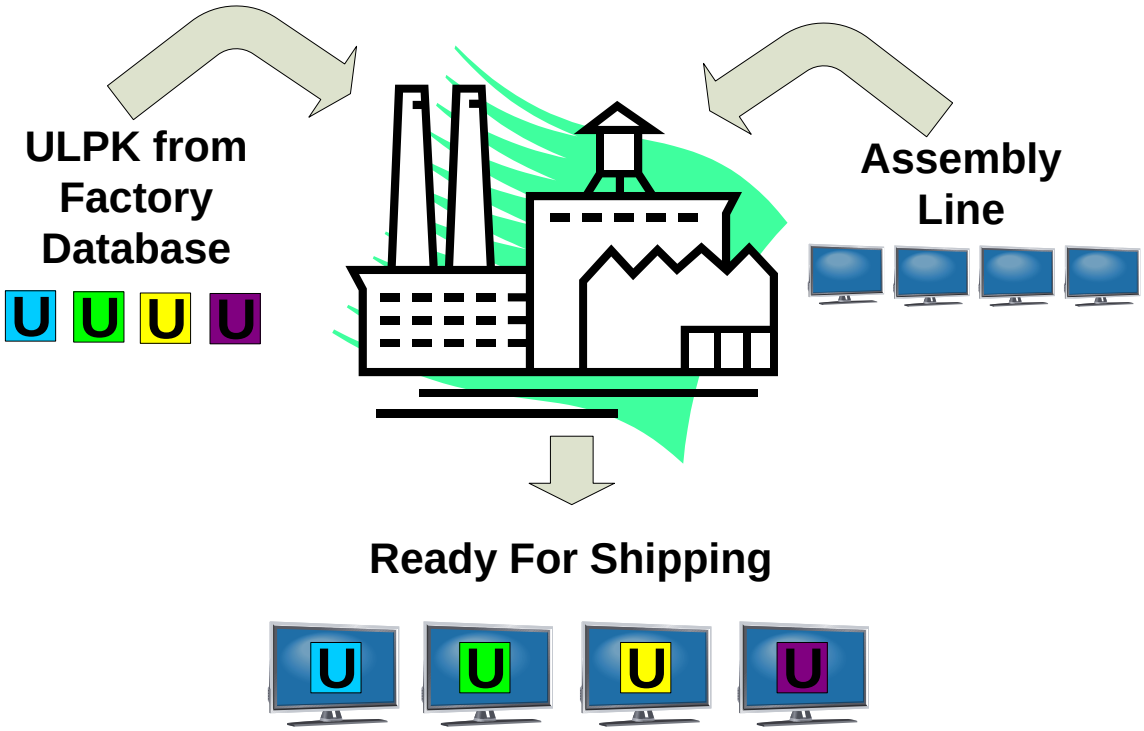
2.1 ULPKs, Serial Number Range Management, and Registration

For a connected device to perform any functions that rely on netReady including OOB testing the device must first "register" with the ULI NOC. In order to register the following must be true:

- A device-unique UpdateLogic Provisioning Key (ULPK) must have been inserted at the factory.
- The serial number of the TV must have been enabled to register on the ULI NOC.
- The device must have an operating connection to the internet.

3.1.1 Understanding the ULPK

Prior to the availability of netProvision, a CEM needed to insert device-unique keys in the factory for each streaming media service that was supported by the device. If there were 4 CSPs in a device there had to be 4 sets of keys inserted at the factory. With netProvision provisioning of these keys occurs in the consumer's home. The factory only relates with a single small, easy to manage ULPK. Each ULPK is device unique and pre-shared with the ULI NOC. The ULPK is used by the Agent during registration so that the NOC can authenticate the device.



ULI generates ULPKs, encrypts them using SoC-specific instructions which means they can only be decrypted on the that particular SoC, and uses base64 encoding to create a CSV file that contains hundreds of thousands or even millions of keys. ULI initially sends the factory a ULPK CSV file with about six months worth of keys. The actual number of keys is dependent on the CEM's production estimates. The first CSV file is delivered 90 days in advance of production. Whenever a factory consumes enough keys so that there are only 90 days of keys left in the CSV "reservoir" then the factory must inform ULI and another CSV file will be sent to top off the reservoir.

The CSV file takes the following format. There are commas between each field and CR/LF between each record.

Index	Key	CRC
1717	QgEBAJ0ZAAAQgSAAMAAAADDXY70emLBRZBHzceJYWi6oeHp5IuHiKSVlb5KdxatPx7ttniLLzDxea6AwFG9z+qYyY+IQ0rTHp3WajxMzsI6Erwl	0x5FE43D77
1718	QgEBAJ0ZAAAQgSAAMAAAAEzthLlqYRaeXZdaCVuAXWXkGs330PonS6CQNJKo7btoe1Zob2MQk54nVkvFxRupM3Cp3b52XsPM1GQYJrlmfIO	0x431428BE
...	...	...

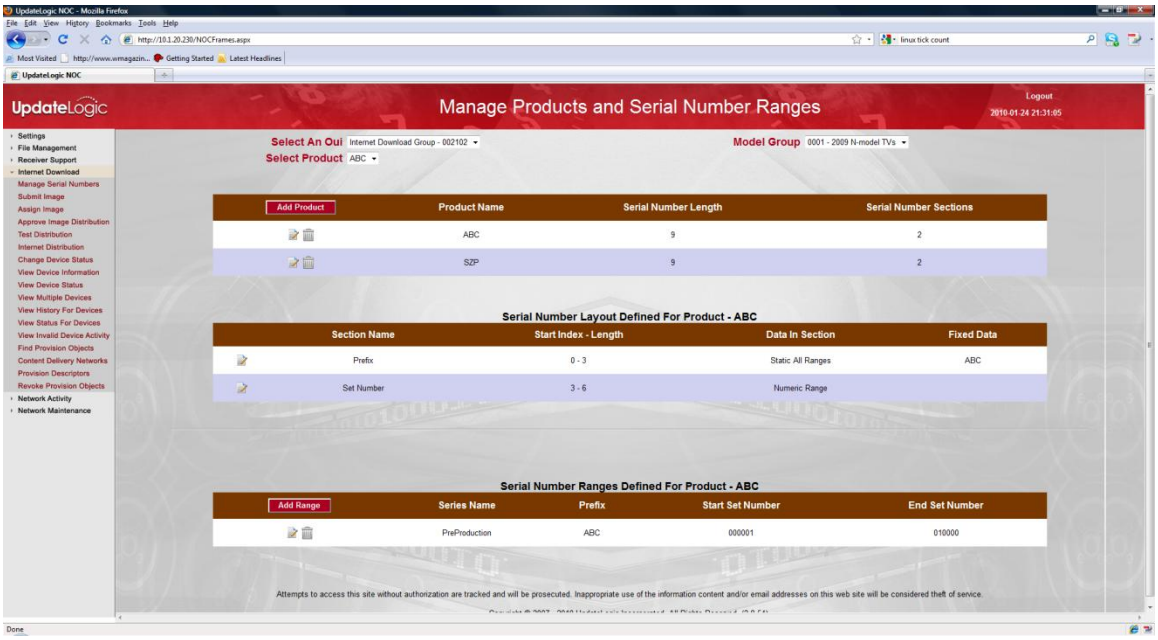
ULI is will provide an example ULPK CSV file upon request.

Each ULPK is extracted by the factory, base64 decoded, and inserted as a file into a location in flash that it known to the Agent though the Agent project adaptation layer. Please see the *UpdateTV Agent Quick Start Guide* for more information about how to specify the location of the ULPK file to the Agent. The factory insertion software should perform a CRC check on each index and key as they are inserted to check CSV file integrity. The factory insertion should read the ULPK back from the device and compare it to the source key to check flash write integrity.

ULPKs do not need to be associated with a particular device serial number. ULPKs may be skipped and discarded. ULPKs should *never* be re-used.

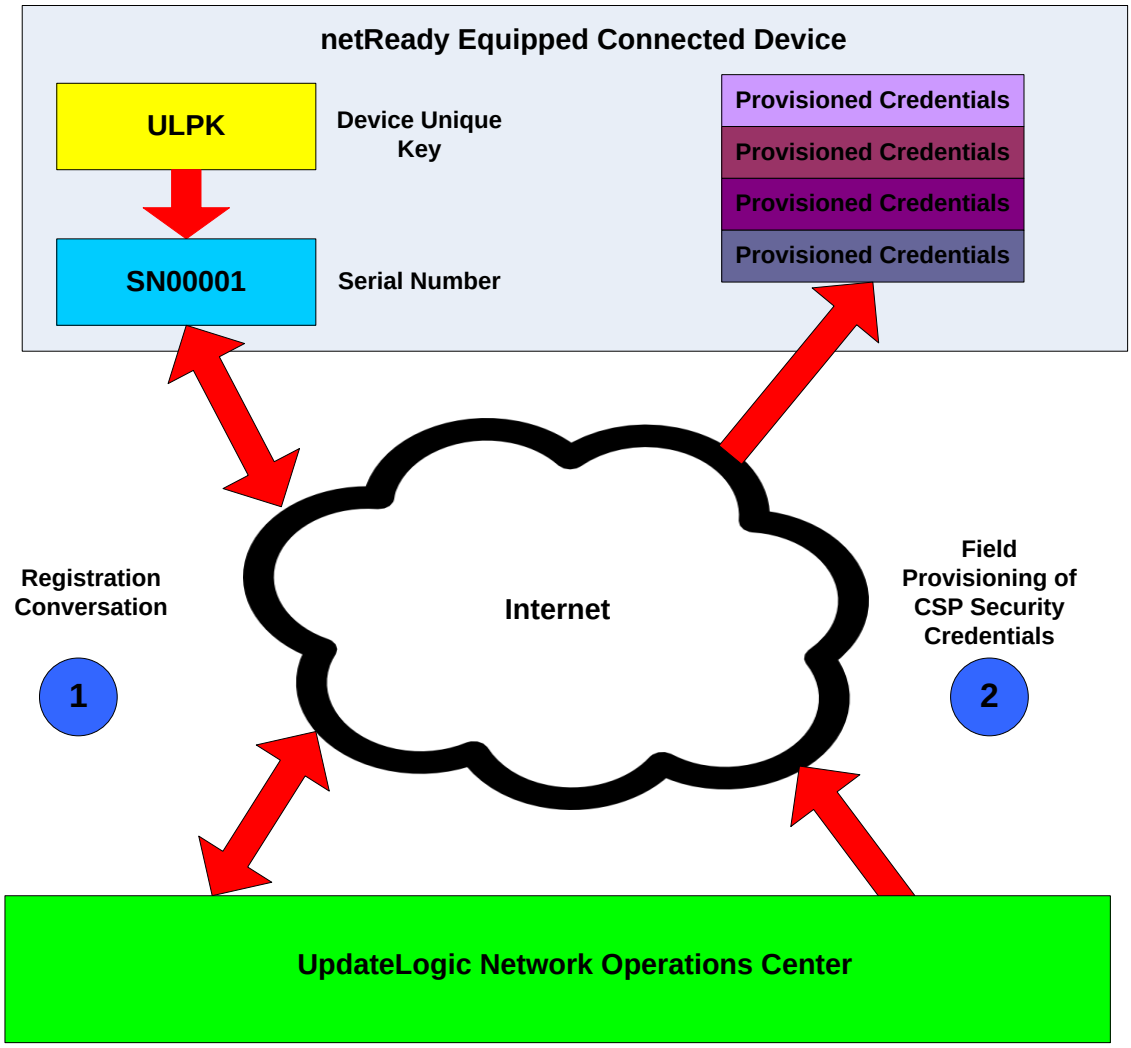
3.1.2 Enabling Serial Numbers on the ULI NOC

In order for a device to be able to register a serial number range must be created and enabled on the ULI NOC. Typically factory serial numbers consist of an alphanumeric prefix, some sort of date stamp, and a serial counting value: ACMETV1200001. The NOC has a powerful serial number configuration page which allows the factory to create serial number ranges that reflect the static and dynamic parts of any serial number format. *Please note:* if a serial number range hasn't been created on the NOC for a given connected device then that device will not be able to register and will not be field provisioned.



3.1.3 Registration

During registration the connected device sends it serial number and its ULPK to the UpdateLogic NOC where they are authenticated. This can take place either in the factory during OOB testing or in the consumer's home. After the device is registered it receives its provisioned credentials from the NOC. These credentials are used by streaming media services to authenticate the device for a given streaming service.



3 Out Of Box Testing

Out of box (OOB) testing has the following goals when applied to connected devices:

- Simulate the end-user experience to test that all manufacturing-related dependencies such as streaming services are operating correctly.
- Perform "end-to-end" testing of netUpdate and netProvision to test that all manufacturing-related dependencies are operating correctly.

3.1 netProvision Testing

During OOB testing the factory's QA department will want to perform whatever steps are required to connect the device to the internet and then exercise all connected device functions that utilize netProvision's field delivered credentials. The number of provisioned credentials is dependent on the services your device supports. It is very useful to know this number and check that the expected number of provisioned objects has been delivered to the device. Typically the number of provisioned objects can be checked in a hidden menu. Please see the section below for

information describing what information should be available in main or hidden menus that can be used during OOB testing.

The most powerful OOB netProvision test is to use all of the connected services on the device to provide that they work.

3.2 netUpdate Testing

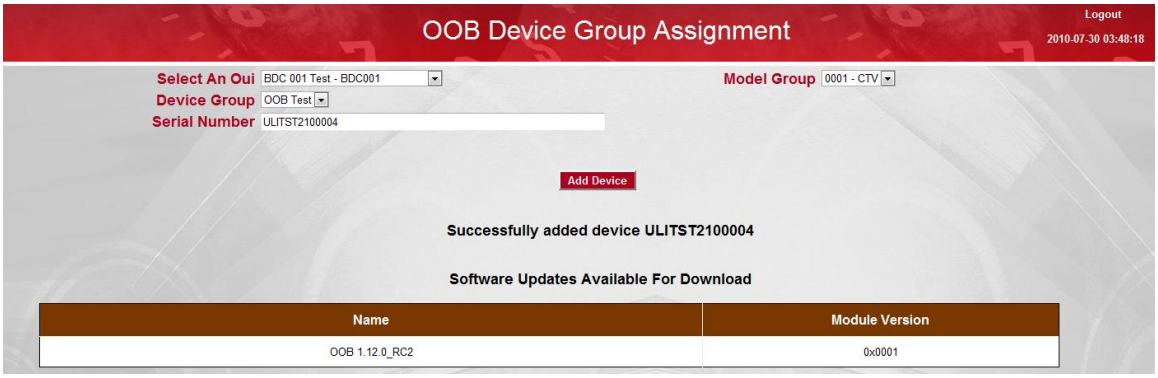
During OOB testing the factory's QA department will also want to test netUpdate. NetUpdate is the service that updates the devices firmware. In order to simplify this testing the OOB test update contains the exact same software that has already been previously gang-programmed or updated into the device on the production line. *It is essential that factory operations staff keep track of the MP version of software and change the OOB software every time the MP software in the device changes.*

3.1.4 Creating the OOB Update

An OOB software update is identical to the MP that is gang-programmed into all manufactured devices. Therefore each gang-programmed software image must be *published* to the NOC as an update that a tested device can then receive. The OOB update test updates the device with the very same software it is already running so that it may be shipped with the OOB update without reverting it.

3.1.5 Performing the OOB Update

In order to perform the OOB update the tester will first have to enter the device's serial number into the OOB Device Group Assignment page of the NOC. Once this is done the device will be put into a special group that is assigned an OOB update. It is best to use a bar code scanner to scan the serial number into the serial number field on the NOC to avoid typing mistakes.



The device may then be turned off or otherwise caused to go into update check and download mode. Update progress may either be displayed in a menu or serial output of debug messages may be monitored.

When the update is complete the device must typically be rebooted so that the update goes into effect. The operator should then check that the updated software has been installed by looking at a ULI version string display message in an appropriate menu.

Once the version is checked nothing more needs to be done to the device. The NOC will automatically remove the device from the OOB device group after an hour.

Here is a summary of the steps above.

1. Add the device to the OOB Device Group Assignment page of the NOC.
2. Power off or otherwise cause the TV to go into update mode.
3. Watch the progress of the update through either a menu or by monitoring serial output.
4. When the update is complete reboot the device and check that the correct ULI version string is displayed.
5. Ship the device (or continue with other OOB testing).

3.3 OSD Support For OOB Testing

Every device that netReady is integrated into should give the user access to certain important information via its on screen display. Some of this information should be easily accessible by the user and some may be placed in hidden menus. This section suggests information that is useful to convey to the user and where it might be located.

Main Help Screen

Electronic Serial Number of the device.

Current TV software version string

Hidden Information

ULI software version number.

ULI module version

ULPK Index

Number of provisioned objects

Device Registered flag

Device in Factory Test Mode flag

Device in Field Test Mode flag

ULI Update Version String

Last Provision Status

Last Download Module Version

Last Download Type

Last Download Status

Last Download Time

Last Error

Last Error Time

Error Count

4 Factory Integration Checklist

This section provides a checklist of action items that the factory needs to accomplish prior to and after mass production begins.

Prior to MP:

1. Factory obtains approximately 12 months of ULPKs from ULI. ULPKs may need to be encrypted with chip-specific hardware key.
2. Factory writes program to import ULPK CSV file into their production database along with items such as serial number and MAC address.
3. Factory works with SoC vendor to make sure device has a factory insertion mode that operates via serial port, Ethernet, or other physical connection to write a unique a ULPK to a known location in flash on the device.
4. Factory tests insertion using hidden menu which displays ULI ULPK index.
5. ULI makes sure that an adequate number of provisioned objects have been imported to the production NOC and that the CSPs have enabled those objects on their servers.
6. Factory optionally obtains lab server to test OOB OAD download.
7. Factory sets up OOB stations with Ethernet cabling or Wi-Fi if supported.
8. Factory holds training seminar at which ULI staff lecture on connected TV logistics, how provisioning and update works.
9. Factory trains OOB testers on correct set of steps for OOB testing.
10. Factory creates process to synchronize RF factory download of MP sw and OOB test update. These sw images must always be the same.
11. Factory assigns staff to enable serial number ranges for the devices that they are making so that OOB registration, provisioning, and update testing work. Initially there should be weekly serial number ranges so that if there are any problems updates to the affected TVs can be tightly targeted. This will require a ULI production NOC log in.
12. Factory speaks to content service providers (CSPs) about "white listing" their factory so that streaming services may be tested outside of the United States.
13. Factory assigns staff to look at daily registration reports which are helpful in spotting production problems.

Transition to MP:

1. CEM engineering makes sure device passes all CSP and ULI validation testing and that device is "locked down" (secure boot, no serial input, no JTAG, no telnetd, etc) before enabling delivery of production provisioned objects. Production provisioned objects must not be delivered to unsecure devices.

MP:

1. Factory performs OOB tests on anywhere from 1 device an hour to more depending on production speed.
2. Factory may optionally conduct OAD OOB test on a smaller selection of devices.
3. Factory reviews daily registration reports for devices that may be displaying error conditions.
4. Factory maintains good contact with ULI Operations staff to help troubleshoot any issues revealed in the daily registration reports.
5. In case there is a problem with an OOB test factory keeps the ability to capture a serial log of device output and sends the log to ULI.

Running Changes:

1. Factory works to closely with CEM engineering and ULI to coordinate new MP sw running changes, RF factory update, OOB updates, and field updates. It's recommended that serial number ranges be used to coordinate cut over to a new sw version. For instance, all devices starting production in week X get the new MP image and the new OOB update is only enabled for that serial number range.

5 Using ULI Daily Reports to Monitor Factory Quality

Every day a report will be generated for each model group that contains connected devices in your OUI. This report shows all of the devices that registered and all of the devices that had problems registering. This report is very useful in tracking factory quality, making sure that expected results are occurring, and becoming alerted to anomalistic behavior.

An example report follows.

Errors

OUI	Model Group	Serial Number	Hit Count	IP Address	Location	Software Version	Hardware Model	Days On Report
0xA12B	1	ABC3900103	1	60.248.88.106	Factory Town Montana	0x79	1	1
0xA12B	1	ABC4000179	1	60.248.88.106		0x64	1	1
0xA12B	1	ABC4000179	1	60.248.88.106		0x20	1	1
0xA12B	1	ABC4000179	1	60.250.89.181		0x79	1	1
0xA12B	1	ABC4000111	1	70.168.249.194	Winchester, MA	0x27	3	11
0xA12B	1	ABC3900007	322	1.2.3.4	Factory Town Alabama	0x3c	1	4
0xA12B	1	ABC3900042	141	1.2.3.4		0x3c	1	4
0xA12B	1	ABC3900046	217	1.2.3.4		0x3c	1	4
0xA12B	1	ABC3900136	729	1.2.3.4		0x3c	1	4
0xA12B	1	ABC3900156	141	1.2.3.4		0x3c	1	4
0xA12B	1	ABC4400008	23	1.2.3.4		0x79	2	2
0xA12B	1	ABC3500095	3	1.2.3.4		0x79	3	2
0xA12B	1	ABC4400022	20	1.2.3.4		0x79	3	2
0xA12B	1	ABC4000179	1	210.80.88.2	Taipei, Taiwan	0x64	1	1
0xA12B	1	ABC3900103	2	221.224.45.100	China	0x79	1	1
0xA12B	1	unknown	4	221.224.45.100		0x79	1	2

Success counts by IP

IP Address	Number of Registrations
1.2.3.4	67
67.167.33.90	1

Successful Registrations

IP Address	Location	OUI	Model Group	Serial Number	Module Version	Hardware Model	Software Version	ULPK
67.167.33.90	Chicago, IL	0xA12B	1	ABC4800123	0x3d	1	0x28	2321
1.2.3.4	Factory, China	0xA12B	1	ABC2100120	0x49	2	0x33	12776
1.2.3.4		0xA12B	1	ABC2100151	0x0	2	0x36	12136
1.2.3.4		0xA12B	1	ABC2100322	0x0	2	0x36	12875
1.2.3.4		0xA12B	1	ABC2100333	0x49	2	0x33	12717
1.2.3.4		0xA12B	1	ABC2100340	0x49	2	0x33	12719
1.2.3.4		0xA12B	1	ABC2100358	0x49	2	0x33	12658
1.2.3.4		0xA12B	1	ABC2100359	0x49	2	0x33	12659
1.2.3.4		0xA12B	1	ABC2100360	0x49	2	0x33	12661
1.2.3.4		0xA12B	1	ABC2100361	0x49	2	0x33	12662
1.2.3.4		0xA12B	1	ABC2100362	0x49	2	0x33	12663
1.2.3.4		0xA12B	1	ABC2100363	0x49	2	0x33	12666
1.2.3.4		0xA12B	1	ABC2100372	0x49	2	0x33	12718
1.2.3.4		0xA12B	1	ABC2100373	0x49	2	0x33	12720
1.2.3.4		0xA12B	1	ABC2100382	0x49	2	0x33	12858
1.2.3.4		0xA12B	1	ABC2100408	0x49	2	0x33	12752
1.2.3.4		0xA12B	1	ABC2100409	0x49	2	0x33	12753
1.2.3.4		0xA12B	1	ABC2100410	0x49	2	0x33	12754
1.2.3.4		0xA12B	1	ABC2100411	0x49	2	0x33	12755
1.2.3.4		0xA12B	1	ABC2100412	0x49	2	0x33	12756
1.2.3.4		0xA12B	1	ABC2100413	0x49	2	0x33	12757
1.2.3.4		0xA12B	1	ABC2100450	0x49	2	0x33	12839
1.2.3.4		0xA12B	1	ABC2100451	0x49	2	0x33	12840

IP Address	Location	OUI	Model Group	Serial Number	Module Version	Hardware Model	Software Version	ULPK
1.2.3.4		0xA12B	1	ABC2100452	0x0	2	0x36	12841
1.2.3.4		0xA12B	1	ABC2100453	0x49	2	0x33	12842
1.2.3.4		0xA12B	1	ABC2100454	0x49	2	0x33	12843
1.2.3.4		0xA12B	1	ABC2100455	0x49	2	0x33	12844
1.2.3.4		0xA12B	1	ABC2100465	0x49	2	0x33	12715
1.2.3.4		0xA12B	1	ABC2100466	0x49	2	0x33	12716
1.2.3.4		0xA12B	1	ABC2100469	0x49	2	0x33	13167
1.2.3.4		0xA12B	1	ABC2100470	0x49	2	0x33	12787
1.2.3.4		0xA12B	1	ABC2100480	0x49	2	0x33	12784
1.2.3.4		0xA12B	1	ABC2100481	0x49	2	0x33	12785
1.2.3.4		0xA12B	1	ABC2100482	0x49	2	0x33	12786
1.2.3.4		0xA12B	1	ABC2100483	0x49	2	0x33	13076
1.2.3.4		0xA12B	1	ABC2100484	0x49	2	0x33	12788
1.2.3.4		0xA12B	1	ABC2100485	0x49	2	0x33	12789
1.2.3.4		0xA12B	1	ABC2100536	0x49	2	0x33	12896
1.2.3.4		0xA12B	1	ABC2100537	0x49	2	0x33	12897
1.2.3.4		0xA12B	1	ABC2100538	0x49	2	0x33	12898
1.2.3.4		0xA12B	1	ABC2100540	0x49	2	0x33	13077
1.2.3.4		0xA12B	1	ABC2100542	0x49	2	0x33	12922
1.2.3.4		0xA12B	1	ABC2100544	0x0	2	0x36	12924
1.2.3.4		0xA12B	1	ABC2100582	0x49	2	0x33	12921
1.2.3.4		0xA12B	1	ABC2100628	0x0	2	0x36	12925
1.2.3.4		0xA12B	1	ABC2100630	0x49	2	0x33	12927
1.2.3.4		0xA12B	1	ABC2100631	0x49	2	0x33	12928
1.2.3.4		0xA12B	1	ABC2100664	0x49	2	0x33	13071
1.2.3.4		0xA12B	1	ABC2100665	0x49	2	0x33	13072
1.2.3.4		0xA12B	1	ABC2100668	0x49	2	0x33	13075
1.2.3.4		0xA12B	1	ABC2100701	0x49	2	0x33	13014
1.2.3.4		0xA12B	1	ABC2100703	0x49	2	0x33	13015
1.2.3.4		0xA12B	1	ABC2100705	0x49	2	0x33	13019
1.2.3.4		0xA12B	1	ABC2100706	0x49	2	0x33	13016
1.2.3.4		0xA12B	1	ABC2100707	0x49	2	0x33	13017

IP Address	Location	OUI	Model Group	Serial Number	Module Version	Hardware Model	Software Version	ULPK
1.2.3.4		0xA12B	1	ABC2100709	0x49	2	0x33	13020
1.2.3.4		0xA12B	1	ABC2100755	0x49	2	0x33	13043
1.2.3.4		0xA12B	1	ABC2100756	0x49	2	0x33	13044
1.2.3.4		0xA12B	1	ABC2100757	0x49	2	0x33	13045
1.2.3.4		0xA12B	1	ABC2100758	0x49	2	0x33	13046
1.2.3.4		0xA12B	1	ABC2100759	0x49	2	0x33	13047
1.2.3.4		0xA12B	1	ABC2100816	0x49	2	0x33	13165
1.2.3.4		0xA12B	1	ABC2100817	0x49	2	0x33	13168
1.2.3.4		0xA12B	1	ABC2100852	0x49	2	0x33	13219
1.2.3.4		0xA12B	1	ABC2100912	0x49	2	0x33	13218
1.2.3.4		0xA12B	1	ABC2100914	0x49	2	0x33	13220
1.2.3.4		0xA12B	1	ABC2100916	0x49	2	0x33	13221
1.2.3.4		0xA12B	1	ABC2100919	0x49	2	0x33	13223

6 Example Code To Read a ULPK CSV File

The following code can be used to create an import utility that extracts ULPKs from their distributed file and imports them into the factory database.

```
1.  /* extract-
    ulpk.c : A sample program to extract a ULPK from a CSV file containing base64 en
    coded ULPKs.
2.
3.     Copyright (c) 2010 UpdateLogic Incorporated. All rights reserved.
4.
5.     Revision History (newest edits added to the top)
6.
7.     Who             Date        Edit
8.     Bob Taylor      02/08/2010  Creation.
9.  */
10.
11. #include <stdio.h>
12. #include <stdlib.h>
13. #include <string.h>
14. #include <signal.h>
15.
16. #define UTV_ULPK_LENGTH 96
17.
18. /* This is the CRC32 table generates with the "forward" polynomial (0x04c11db7)

19. */
20. static const unsigned long UtvCrc32Table[ 256 ] =
21. {
```

```
    0x00000000UL, 0x04c11db7UL, 0x09823b6eUL, 0x0d4326d9UL,
    0x130476dcUL, 0x17c56b6bUL, 0x1a864db2UL, 0x1e475005UL,
    0x2608edb8UL, 0x22c9f00fUL, 0x2f8ad6d6UL, 0x2b4bcb61UL,
    0x350c9b64UL, 0x31cd86d3UL, 0x3c8ea00aUL, 0x384fbdbdUL,
    0x4c11db70UL, 0x48d0c6c7UL, 0x4593e01eUL, 0x4152fda9UL,
    0x5f15adacUL, 0x5bd4b01bUL, 0x569796c2UL, 0x52568b75UL,
    0x6a1936c8UL, 0x6ed82b7fUL, 0x639b0da6UL, 0x675a1011UL,
    0x791d4014UL, 0x7ddc5da3UL, 0x709f7b7aUL, 0x745e66cdUL,
    0x9823b6e0UL, 0x9ce2ab57UL, 0x91a18d8eUL, 0x95609039UL,
    0x8b27c03cUL, 0x8fe6dd8bUL, 0x82a5fb52UL, 0x8664e6e5UL,
    0xbe2b5b58UL, 0xbaea46efUL, 0xb7a96036UL, 0xb3687d81UL,
    0xad2f2d84UL, 0xa9ee3033UL, 0xa4ad16eaUL, 0xa06c0b5dUL,
    0xd4326d90UL, 0xd0f37027UL, 0xddb056feUL, 0xd9714b49UL,
    0xc7361b4cUL, 0xc3f706fbUL, 0xceb42022UL, 0xca753d95UL,
    0xf23a8028UL, 0xf6fb9d9fUL, 0xfbb8bb46UL, 0xff79a6f1UL,
    0xe13ef6f4UL, 0xe5ffeb43UL, 0xe8bccd9aUL, 0xec7dd02dUL,
    0x34867077UL, 0x30476dc0UL, 0x3d044b19UL, 0x39c556aeUL,
    0x278206abUL, 0x23431b1cUL, 0x2e003dc5UL, 0x2ac12072UL,
    0x128e9dcfUL, 0x164f8078UL, 0x1b0ca6a1UL, 0x1fcdbb16UL,
    0x018aeb13UL, 0x054bf6a4UL, 0x0808d07dUL, 0x0cc9cdcaUL,
    0x7897ab07UL, 0x7c56b6b0UL, 0x71159069UL, 0x75d48ddeUL,
    0x6b93dddbUL, 0x6f52c06cUL, 0x6211e6b5UL, 0x66d0fb02UL,
    0x5e9f46bfUL, 0x5a5e5b08UL, 0x571d7dd1UL, 0x53dc6066UL,
    0x4d9b3063UL, 0x495a2dd4UL, 0x44190b0dUL, 0x40d816baUL,
    0xaca5c697UL, 0xa864db20UL, 0xa527fdf9UL, 0xa1e6e04eUL,
    0xbfa1b04bUL, 0xbb60adfcUL, 0xb6238b25UL, 0xb2e29692UL,
    0x8aad2b2fUL, 0x8e6c3698UL, 0x832f1041UL, 0x87ee0df6UL,
    0x99a95df3UL, 0x9d684044UL, 0x902b669dUL, 0x94ea7b2aUL,
    0xe0b41de7UL, 0xe4750050UL, 0xe9362689UL, 0xedf73b3eUL,
    0xf3b06b3bUL, 0xf771768cUL, 0xfa325055UL, 0xfef34de2UL,
    0xc6bcf05fUL, 0xc27dede8UL, 0xcf3ecb31UL, 0xcbffd686UL,
    0xd5b88683UL, 0xd1799b34UL, 0xdc3abdedUL, 0xd8fba05aUL,
    0x690ce0eeUL, 0x6dcdfd59UL, 0x608edb80UL, 0x644fc637UL,
    0x7a089632UL, 0x7ec98b85UL, 0x738aad5cUL, 0x774bb0ebUL,
    0x4f040d56UL, 0x4bc510e1UL, 0x46863638UL, 0x42472b8fUL,
    0x5c007b8aUL, 0x58c1663dUL, 0x558240e4UL, 0x51435d53UL,
    0x251d3b9eUL, 0x21dc2629UL, 0x2c9f00f0UL, 0x285e1d47UL,
    0x36194d42UL, 0x32d850f5UL, 0x3f9b762cUL, 0x3b5a6b9bUL,
    0x0315d626UL, 0x07d4cb91UL, 0x0a97ed48UL, 0x0e56f0ffUL,
    0x1011a0faUL, 0x14d0bd4dUL, 0x19939b94UL, 0x1d528623UL,
    0xf12f560eUL, 0xf5ee4bb9UL, 0xf8ad6d60UL, 0xfc6c70d7UL,
    0xe22b20d2UL, 0xe6ea3d65UL, 0xeba91bbcUL, 0xef68060bUL,
    0xd727bbb6UL, 0xd3e6a601UL, 0xdea580d8UL, 0xda649d6fUL,
    0xc423cd6aUL, 0xc0e2d0ddUL, 0xcda1f604UL, 0xc960ebb3UL,
    0xbd3e8d7eUL, 0xb9ff90c9UL, 0xb4bcb610UL, 0xb07daba7UL,
    0xae3afba2UL, 0xaafbe615UL, 0xa7b8c0ccUL, 0xa379dd7bUL,
    0x9b3660c6UL, 0x9ff77d71UL, 0x92b45ba8UL, 0x9675461fUL,
    0x8832161aUL, 0x8cf30badUL, 0x81b02d74UL, 0x857130c3UL,
    0x5d8a9099UL, 0x594b8d2eUL, 0x5408abf7UL, 0x50c9b640UL,
    0x4e8ee645UL, 0x4a4ffbf2UL, 0x470cdd2bUL, 0x43cdc09cUL,
    0x7b827d21UL, 0x7f436096UL, 0x7200464fUL, 0x76c15bf8UL,
    0x68860bfdUL, 0x6c47164aUL, 0x61043093UL, 0x65c52d24UL,
    0x119b4be9UL, 0x155a565eUL, 0x18197087UL, 0x1cd86d30UL,
    0x029f3d35UL, 0x065e2082UL, 0x0b1d065bUL, 0x0fdc1becUL,
    0x3793a651UL, 0x3352bbe6UL, 0x3e119d3fUL, 0x3ad08088UL,
    0x2497d08dUL, 0x2056cd3aUL, 0x2d15ebe3UL, 0x29d4f654UL,
    0xc5a92679UL, 0xc1683bceUL, 0xcc2b1d17UL, 0xc8ea00a0UL,
    0xd6ad50a5UL, 0xd26c4d12UL, 0xdf2f6bcbUL, 0xdbee767cUL,
    0xe3a1cbc1UL, 0xe760d676UL, 0xea23f0afUL, 0xeee2ed18UL,
    0xf0a5bd1dUL, 0xf464a0aaUL, 0xf9278673UL, 0xfde69bc4UL,
    0x89b8fd09UL, 0x8d79e0beUL, 0x803ac667UL, 0x84fbdbd0UL,
```

```
    0x9abc8bd5UL, 0x9e7d9662UL, 0x933eb0bbUL, 0x97ffad0cUL,
    0xafb010b1UL, 0xab710d06UL, 0xa6322bdfUL, 0xa2f33668UL,
    0xbcb4666dUL, 0xb8757bdaUL, 0xb5365d03UL, 0xb1f740b4UL
};

/* UtvCrc32
    Forward CRC-32 polynomial use generator 0x04C11DB7
    x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x^1+x^0
*/
unsigned long UtvCrc32( const unsigned char * pBuffer, unsigned long uiSize )
{
    unsigned long uiCrc = 0xffffffff;
    const unsigned char * p;

    for ( p = pBuffer; uiSize > 0; ++p, --uiSize )
        uiCrc = (uiCrc << 8) ^ UtvCrc32Table[(uiCrc >> 24) ^ *p];
    return uiCrc;
}


void usage( )
{
    printf( "extract-ulpk\n" );
    printf( "usage: extract-
ulpk <ULPK output file> <Index> <ulpk CSV file>\n" );
}

/*
** Translation Table as described in RFC1113
*/
static const char cb64[]="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01
23456789+/";

/*
** Translation Table to decode (created by author)
*/
static const char cd64[]="|$$$}rstuvwxyz{$$$$$$$>?@ABCDEFGHIJKLMNOPQRSTUVW$$$$$$
XYZ[\\]^_`abcdefghijklmnopq";

/*
** decodeblock
**
** decode 4 '6-bit' characters into 3 8-bit binary bytes
*/
static void decodeblock( unsigned char in[4], unsigned char out[3] )
{
    out[ 0 ] = (unsigned char ) (in[0] << 2 | in[1] >> 4);
    out[ 1 ] = (unsigned char ) (in[1] << 4 | in[2] >> 2);
    out[ 2 ] = (unsigned char ) (((in[2] << 6) & 0xc0) | in[3]);
}

/*
** decode
**
** decode a base64 encoded stream discarding padding, line breaks and noise
*/
static void decode( char *inbuff, unsigned long InLen, unsigned char *outbuff, u
nsigned long *puiOutLen )
{
    unsigned char in[4], out[3], v;
    unsigned long i, len, n = 0, o = 0;
```

```
140.
141.    while( n < InLen )
142.    {
143.        for( len = 0, i = 0; i < 4 && n < InLen; i++ ) {
144.            v = 0;
145.            while( n < InLen && v == 0 ) {
146.                v = (unsigned char) inbuff[ n++ ];
147.                v = (unsigned char) ((v < 43 || v > 122) ? 0 : cd64[ v - 43 ]);
148.                if( v ) {
149.                    v = (unsigned char) ((v == '$') ? 0 : v - 61);
150.                }
151.            }
152.            if( n < InLen ) {
153.                len++;
154.                if( v ) {
155.                    in[ i ] = (unsigned char) (v - 1);
156.                }
157.            }
158.            else {
159.                in[i] = 0;
160.            }
161.        }
162.        if( len ) {
163.            decodeblock( in, out );
164.            for( i = 0; i < len - 1; i++ ) {
165.                outbuff[ o++ ] = out[i];
166.            }
167.        }
168.    }
169.
170.    *puiOutLen = o;
171.}
172.
173./*
174.** Open the passed in CSV and attempt to extract the ULPK with the given index.
175.*/
176.int UtvRetrieveULPKFromCSV( char *pszFName, unsigned long uiTargetIndex, unsigned char *ulpkBuff, unsigned long *puiLen )
177.{
178.    unsigned long   ulpkLen, uiDataLen, uiCount = 0, uiIndex = uiTargetIndex, uiCRC;
179.    char            cBuff[ 1024 ], *index, *ulpk, *csum;
180.    int             bFirstTime = 1, bSkipping = 0, bFound = 0;
181.    FILE           *fp;
182.
183.    /* open the CSV file */
184.    if ( NULL == ( fp = fopen( pszFName, "r" )))
185.    {
186.        printf( "UtvPlatformFileOpen(%s) fails.\n", pszFName );
187.        return 1;
188.    }
189.
190.    while ( NULL != fgets( cBuff, sizeof(cBuff), fp ))
191.    {
192.        /* attempt to read in a line of the CSV */
193.        index = strtok( cBuff, "," );
194.        ulpk = strtok( NULL, "," );
195.        csum = strtok( NULL, "," );
196.
```

```
197.        if ( uiIndex != strtoul( index, NULL, 10 ) )
198.        {
199.            if ( bFirstTime )
200.            {
201.                printf( "Skipping...looking for %u\n", uiIndex );
202.                bFirstTime = 0;
203.                bSkipping = 1;
204.            }
205.
206.            if ( bSkipping )
207.                continue;
208.
209.            printf( "Tracking index disagrees with CSV index: %d != %s\n", uiInd
    ex, index );
210.            return 0;
211.        }
212.
213.        bFirstTime = 0;
214.
215.        if ( bSkipping )
216.        {
217.            printf( "Skipped ahead to index: %u\n", uiIndex );
218.        }
219.
220.        bSkipping = 0;
221.
222.#ifdef _dbg
223.        printf(  "Index string: %s", index );
224.        printf(  "Encoded ULPK: %s", ulpk );
225.        printf(  "Checksum: %s", csum );
226.#endif
227.
228.        if ( NULL != ulpk )
229.        {
230.            decode( ulpk, strlen( ulpk ) + 1, ulpkBuff, &ulpkLen );
231.        } else
232.        {
233.            printf( "ERROR: NULL ulpk string\n" );
234.            return 0;
235.        }
236.
237.        /* check data len */
238.        uiDataLen = ulpkLen;
239.
240.        if ( UTV_ULPK_LENGTH != ulpkLen )
241.        {
242.            printf( "ERROR: decoded ULPK from CSV isn't %d bytes in length: %d\n
    ", UTV_ULPK_LENGTH, ulpkLen );
243.            return 0;
244.        }
245.
246.        *puiLen = uiDataLen;
247.
248.        uiCRC = strtoul( csum, NULL, 0 );
249.
250.        /* calculate the encrypted CRC on the ULPK and check it */
251.        if ( uiCRC != UtvCrc32( (const unsigned char *) ulpkBuff, ulpkLen ))
252.        {
253.            printf( "Bad CRC on encrypted ULPK with index %u.  Stamped CRC: 0x%0
    8X\n", uiIndex, uiCRC );
254.            return 0;
```

```
255.        }
256.
257.        bFound = 1;
258.        break;
259.
260.    }
261.
262.    fclose( fp );
263.
264.    if ( bFound )
265.    {
266.        printf( "Target ULPK with index %u found\n", uiTargetIndex );
267.        return 1;
268.    }
269.
270.    printf( "Target ULPK with index %u NOT found!\n", uiTargetIndex );
271.    return 0;
272.}
273.
274./* Main program
275.*/
276.int main( int argc, char * argv[] )
277.{
278.    FILE              *f;
279.    unsigned long     iIndex = 0, uiLen;
280.    unsigned char     aubULPKBuff[256];
281.
282.    if ( 4 != argc )
283.    {
284.        usage();
285.        exit( 1 );
286.    }
287.
288.    /* Announce program name */
289.    printf( "extract-ulpk\n" );
290.
291.    /* Open the ULPK file for write. */
292.    if ( NULL == (f = fopen( argv[ 1 ], "wb") ))
293.    {
294.        printf( "Unable to open file \"%s\" for write\n", argv[ 1 ] );
295.        exit( 1 );
296.    }
297.
298.    /* get index */
299.    iIndex = strtoul( argv[ 2 ], NULL, 10 );
300.
301.    /* extract the ulpk */
302.    if ( !UtvRetrieveULPKFromCSV( argv[ 3 ], iIndex, aubULPKBuff, &uiLen ))
303.    {
304.        printf( "UtvRetrieveULPKFromCSV error\n" );
305.        return 1;
306.    }
307.
308.    if ( uiLen != fwrite( aubULPKBuff, 1, uiLen, f ) )
309.    {
310.        printf( "Unable to write ULPK index.  Out of disk space?\n" );
311.        return 1;
312.    }
313.
314.    fclose( f );
315.
```

```
316.    printf( "All done.\n" );
317.
318.    return 0;
319.}
```