DIMARK SOFTWARE, INC.

Dimark Embedded Client

v.4.0

TECHNICAL REFERENCE & IMPLEMENTATION GUIDE

February, 2011

A LASER FOCUS ON SECURE WAN MANAGEMENT

Contents

Legal Notices

Copyright Notice

Warranty Disclaimer

THE DIMARK TECHNOLOGY AND ANY ACCOMPANYING MATERIALS AND INFORMATION ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND DIMARK EXPLICITLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.

License

PLEASE READ THIS LICENSE CAREFULLY BEFORE USING THE SOFTWARE. BY USING THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE. DIMARK SOFTWARE, INC. ("DIMARK") SOFTWARE IS LICENSED NOT SOLD.

FOR WARRANTY INFORMATION PERTAINING TO THIS PRODUCT, PLEASE REFER TO THE WARRANTY DISCLAIMER ABOVE.

1. License. The application, demonstration, system and other software accompanying this License, whether on disk, in read-only memory, or on any other media (the "Dimark Software"), and the related documentation are licensed to you by Dimark. You own the medium on which the Dimark Software are recorded, but Dimark and /or Dimark's licensor(s) retain title to the Dimark Software and related documentation. The License allows you to use the Dimark Software on a single Dimark product and make one copy of the Dimark Software in machine-readable form only for backup purposes. You must reproduce, on such copy, the Dimark copyright notice and any other proprietary legends that were on the original copy of the Dimark Software. You may also transfer all your license rights in the Dimark Software, the backup copy of the Dimark Software, the related documentation, and a copy of this License to another party provided the other party reads and agrees to accept the terms and conditions of this License.

2. Restrictions. The Dimark Software contains copyrighted material, trade secrets, and other proprietary material. In order to protect them, and except as permitted by applicable legislation, you may not decompile, reverse engineer, disassemble, or otherwise reduce the Dimark Software to a human-perceivable form: copy, modify, network, rent, lease, loan, or distribute the Dimark Software; or create derivative works based upon the Dimark Software in whole or in part. You may not electronically transmit the Dimark Software from one computer to another or over the network.

3. Termination. This License is effective until terminated. You may terminate the License at any time by destroying the Dimark Software, related documentation and all copies thereof. This License will terminate immediately without notice from Dimark if you fail to comply with any provision of this License. Upon termination you must destroy the Dimark Software, related documentation, and all copies thereof. Upon termination you shall remain subject to the provisions, restrictions and exclusions in this License Agreement and shall have no right to any refund of any amount paid for the Dimark Software. No termination shall release you from liability for any breach of this License Agreement.

4. Export Law Assurances. You agree and certify that neither the Dimark Software nor any other technical data received from Dimark, nor the direct product thereof, will be shipped, transferred, or exported, directly or indirectly, to any county in violation of any applicable law, including the United States Export Administration Act and the regulations there under.

5. Controlling Law and Severability. This License shall be governed by and construed in accordance with the laws of the State of California without regard to its conflict of laws provisions. If for any reason a court of competent jurisdiction finds any provision of this License, or portion thereof, to be unenforceable, that provision of the License shall be enforced to the maximum extent permissible so as to affect the intent of the parties, and the remainder of this License shall continue in full force and effect.

6. Acknowledgment. You acknowledge that you have read this License Agreement, understand it, and agree to be bound by its terms and conditions. You also agree that the License agreement is the complete and exclusive statement of agreement between the parties and supersedes all proposals or prior agreements, oral or written, and any other communications between the parties relating to the subject matter of the License Agreement. No amendment to or modification of this License will be binding unless in writing and signed by a duly authorized representative of Dimark.

1. Dimark TR-069 Client Integration

Version 4.0

This document describes how to integrate the client into a build system and performing extensions to the client. It also describes how to integrate the application into the host system so that notification events can dynamically be forwarded to the Auto configuration server (ACS).

The base client provided by Dimark provides a configuration file that defines some generic TR-069 parameters that identify the device and how to communicate with the ACS. All other functionality needs to be provided during the integration/implementation phase. This allows the client to operate with virtually any host system available.

2. About Dimark's Embedded TR-069 Client

The Dimark TR-069 client provides a complete TR-069 implementation, including, but not limited to TR-069, TR-098, TR-104, TR-106, TR-110, TR-111, TR-135, TR-140, TR-142, TR-143 and TR-196, as well as other third party specifications, such as WiMax Forum Specification for TR-069 devices.

The key functions offered by these protocols is as follows:

- TR-069 enables an extensible, secure, communications layer, while also providing basic gateway router and Wi-Fi configuration and management functionality [1].
- TR-098 enables QoS functionality and provides configuration profiles to ease management and deployment of gateway devices [2].
- TR-104 and TR-110 combine to provide remote VoIP device configuration and management [3, 4].
- TR-106 and TR-111 combine to allow the remote management of devices on a LAN, even those using the private IP space behind a NAT gateway [5, 6].
- TR-135 enables the configuration and management of Set Top Boxes (STB) [7].
- TR-140 enables the configuration and management of Network Attached Storage (NAS) [8].
- TR-142 enables the configuration and management of PON devices [9].
- TR-143 enables network throughput performance tests and statistical monitoring [10].
- TR-157 enables component objects for CWMP [11].
- TR-181 defines data models for Device and Internet Gateway Device [12].
- TR-196 enables support for Femtocell (cellular repeater) devices [13].

The Dimark TR-069 client is typically provided as ANSI C source code that runs on embedded Linux, with the API designed to ease the integration of the client into existing environments. Partners have also quickly deployed the client on non-Linux-based RTOS platforms (VxWorks, Nucleus, etc.), as well as WinCE.

The client incorporates an abstraction layer that will speed implementation as well as make the addition of many new features, upgrades and updates virtual drop-ins.

The client incorporates all updates that have resulted from more than five years of field use, feature requests and customer deployments.

3. New in this Release

- Initial parameters are refactored and read from XML file.
- Host interface can be disabled / enabled via corresponding compiler flag.
- Download and upload functionality was corrected:
  - Callback function is called before a file is downloaded by the CPE.
  - TransferComplete is called only after the transfer was successfully completed, and in the particular cases when the downloaded file is a software image, TransferComplete is called only after downloaded file was actually installed and is operational.
  - Download is not accepted on empty 'Target name' field when using Dimark ACS 2.3.2 for WebLogic.
- Debug log was refactored.
- Minor editorial changes.

4. Client File Structure

In the Client file structure files are grouped according to purpose. Parts of a Client directory tree are listed below. All directories are grouped under the root directory named after current Client version.

- TR-098 – stores initial client parameter configuration file for Internet Gateway Device Data Model
  - data-model.xml – sample of client parameter configuration file in .xml format
- TR-106 – stores initial client parameter configuration file for TR-069-Enabled Devices
  - data-model.xml – sample of client parameter configuration file in .xml format
- gsoap – stores files related with gSOAP functionality
  - soapcpp2 – gSOAP Version 2.7.6c compilation resulting file
  - soapdefs.h – using this file user can amend gSOAP buffers size
  - stdsoap2.[ch] – files borrowed from gSOAP Version 2.7.6c and customized according to Dimark needs
- plugin – stores files responsible for digest access authentication (Upload, Download and ACS connectivity)
  - httpda.[ch]
  - md5evp.[ch]
  - smdevp.[ch]
  - threads.h
- src
  - ftp – stores files responsible for FTP Downloads and FTP Uploads
    - cmds.[ch]
    - ftp.c
    - ftp_ft.[ch]
    - ftp_var.h
  - handlers – handlers of corresponding entities
    - acshandler.[ch]
    - hosthandler.[ch]
    - kickedhandler.[ch]
    - stunhandler.[ch]
    - timehandler.[ch]
    - transfershandler.[ch]
  - host – stores files that allow to implement basic procedures (parameters and events storage, storage of information needed for Client data recovery in case of its Reboot, etc.)
    - diagParameter.[ch]
    - ethParameter.[ch]
    - eventStore.[ch]
    - filetransferStore.[ch]
    - optionStore.[ch]
    - parameterStore.[ch]

- storage.h
- options – stores files for SetVouchers and GetOptions methods implementation
  - option.[ch]
- tr104 – stores files for operating with voice-over-IP (VoIP) devices
  - voipParameter.[ch]
- tr111 – stores files for operating with STUN protocol
  - stun_client.[ch]
  - stun_dimark.[ch]
  - stun_packet.[ch]
  - stun_util.[ch]
  - TR111_README.txt
- callback.[ch]
- debug.[ch]
- dimclient.[ch]
- eventcode.[ch]
- filetransfer.[ch]
- ftcallback.[ch]
- globals.h
- list.[ch]
- methods.[ch]
- paramaccess.[ch]
- paramconvenient.[ch]
- parameter.[ch]
- serverdata.[ch]
- utils.[ch]
- vouchers.[ch]

- tmp – stores temporary files
- README – stores README files
  - README – main README file containing information of how parameters are handled and how functionality works. Usage of HTTP BasicAuthentication and default parameter file structure are also described in this file
  - README_compile.txt – instructions for compiling Dimark Client on Fedora/Centos/RedHat Linux platform
  - README_sslauth.txt – Dimark Client SSL authentication notes
- ChangeLog*.txt – describes all the changes made to Client in each release version
- config.txt – stores file that will be uploaded if Upload argument FileType is “1 Vendor Configuration File”
- conv-util.c – allows the client to read *.xml files and treat them as parameter configuration files
- conv.jar – converts old format of client parameter configuration files to new one (dps.param to data-model.xml) and vice versa
- clientEditor.jar – allows the user to edit *.param files

- download.dwn – this file is created if the path for file download is not specified (in this case file will be downloaded to download.dwn)
- host-if.c – example for host interface testing
- logfile.txt – stores file that will be uploaded if Upload argument FileType is "2 Vendor Log File"
- Makefile – make utility performs commands according to rules set in this file
- start.sh – starts Dimark Client. See the start.sh script to find out how the Client needs to be started and how its target data directory should be prepared
- DEBUG.log – file generated by gSOAP
- TEST.log – file generated by gSOAP

The client provides the option to modify the way it stores data. The default implementation that the client ships with is using a regular file system which might not be available on the target system.

The src/host directory contains all File I/O related access methods that store data, events, options, etc. This allows updating the code without loosing your File I/O modifications.

The default implementation uses several directories to store files and are defined in src/host/storage.h

Define	Typical Value	Usage
PERSISTENT_PARAMETER_DIR	./tmp/parameter/	Directory where the list of added parameters are stored
PERSISTENT_DATA_DIR	./tmp/data/	Directory where the parameters are stored
PERSISTENT_FILE	./tmp/bootstrap.dat	Persistent client data that must survive a Reboot and a Boot, but not survive FactoryReset
EVENT_STORAGE_FILE	./tmp/eventCode.dat	Event Storage of the client that must survive a reboot.
DEFAULT_DOWNLOAD_FILE	./download.dwn	Default download file
PERSISTENT_OPTION_DIR	./tmp/options/	Directory where TR-069 Option data is stored
VOUCHER_FILE	./tmp/Voucher_%d	Voucher creation file string (files must survive reboot)

The location of those files should be changed to be in line with the file systems used on the CPE.

5. Porting the Client

Before porting the Dimark Client onto your target platform, it is strongly recommended to build the client on a regular Linux machine (i.e. Red Hat Enterprise Linux 5+, SUSE Linux Enterprise 10+, etc) to familiarize yourself with the output of the client and to establish a point of reference before starting the porting effort.

> Starting from Client v. 4.0 in order to run and compile Dimark Client user should check the presence and version of the `crypto`, `ssl` and `xml2` shared libraries.

Please perform the following steps before running or compileing Dimark Client:

1. Run `rpm -qa | grep openssl` command.
It checks the presence of both `ssl` and `crypto` shared libraries.

2. Run `rpm -qa | grep libxml2` command.
It checks the presence of `xml2` shared library.

If user hadn't checked the libraries before compilation he can perform the libraries check when he already has an executable file. To perform the check for `dimclient` file, run the following command:

```
ldd dimclient
```

You will see the results presented in the form of two columns. First column reflects requested library name and version and the second reflects its availability:

```
linux-gate.so.1 =>  (0xffffe000)
libpthread.so.0 => /lib/libpthread.so.0 (0xf775a000)
libcrypto.so.6 => not found <--------> Library is not available
libssl.so.6 => not found<--------> Library is not available
libxml2.so.2 => /usr/lib/libxml2.so.2 (0xf760c000)
libc.so.6 => /lib/libc.so.6 (0xf74a0000)
/lib/ld-linux.so.2 (0xf779c000)
libdl.so.2 => /lib/libdl.so.2 (0xf749b000)
libz.so.1 => /lib/libz.so.1 (0xf7487000)
libm.so.6 => /lib/libm.so.6 (0xf745d000)
```

Compiler Flags for the Client

Please note, for compilation required development versions of the libraries:
`libxml2-devel, openssl-devel`.

In case if the libraries are located in another folder, perform the following command to create a symbolic link:

```
ln -s /usr/include/libxml2/libxml /usr/include/libxml
```

When building your reference system, make sure that compiler flags are set as needed on the target system to create an appropriate simulation.

Please consider that Client memory consumption depends on number of flags you have set. By disabling methods that you are not going to use you can reduce memory consumption up to 50-70%.

Please see below approximate Client memory consumption at worst and at best:

Client memory consumption at worst		Client memory consumption at best	
VmPeak:	72764 kB	VmPeak:	23296 kB
VmSize:	72752 kB	VmSize:	23284 kB
VmLck:	0 kB	VmLck:	0 kB
VmHWM:	2612 kB	VmHWM:	2468 kB
VmRSS:	2604 kB	VmRSS:	2460 kB
VmData:	50064 kB	VmData:	656 kB
VmStk:	136 kB	VmStk:	136 kB
VmExe:	644 kB	VmExe:	584 kB
VmLib:	5248 kB	VmLib:	5248 kB
VmPTE:	84 kB	VmPTE:	60 kB
VmSwap:	0 kB	VmSwap:	0 kB
Threads:	7	Threads:	1

Compiler Flag	Location	Usage
ACS_REGMAN	Makefile	Enable support for the REGMAN ACS. Do NOT enable this option unless the REGMAN ACS is used as it is not fully compliant with regular TR-069.
FILETRANSFER_STORE_STATUS	globals.h	Enables / disables saving information about file transfer. Option is useful if device is rebooted while the file transfer is not completed. In this case saved information is used to resume the transfer.

Compiler Flag	Location	Usage
HANDLE_NUMBERS_OF_ ENTRIES	globals.h	Define if the client should update the NumberOfEntries of an Object during an AddObject or DeleteObject.
HAVE_ATMF5_DIAGNOSTICS	globals.h	Enables / disables ATMF5 diagnostics.
HAVE_AUTONOMOUS_ TRANSFER_COMPLETE	globals.h	Enables / disables AutonomousTransferComplete method.
HAVE_DIAGNOSTICS	globals.h	Enables / disables all diagnostics.
HAVE_DOWNLOAD_ DIAGNOSTICS	globals.h	Enables / disables DownloadDiagnostics method. Does not concern Download method.
HAVE_FACTORY_RESET	globals.h	Enables / disables FactoryReset method.
HAVE_FILE	globals.h	Enables / disables following flags: HAVE_UDP_ ECHO, HAVE_GET_QUEUED_TRANSFERS, HAVE_SCHEDULE_INFORM, HAVE_REQUEST_ DOWNLOAD and TransferComplete method.
HAVE_FILE_DOWNLOAD	globals.h	Enables / disables both Download and DownloadDiagnostics methods.
HAVE_FILE_UPLOAD	globals.h	Enables / disables both Upload and UploadDiagnostics methods.
HAVE_GET_ALL_QUEUED_ TRANSFERS	globals.h	Enables / disables GetAllQueuedTransfers method. If this flag is disabled AutonomousTransferComplete method (HAVE_AUTONOMOUS_TRANSFER_COMPLETE flag) is also automatically disabled.
HAVE_GET_QUEUED_ TRANSFERS	globals.h	Enables / disables GetAllQueuedTransfers method.
HAVE_HOST	globals.h	Enables / disables host interface support.
HAVE_IP_PING_ DIAGNOSTICS	globals.h	Enables / disables IP Ping diagnostics.
HAVE_KICKED	globals.h	Enables / disables Kicked method.
HAVE_OPTIONAL	globals.h	Enables / disables CPE optional responding methods: Download, Upload, FactoryReset, GetQueuedTransfers, GetAllQueuedTransfers, ScheduleInform, SetVouchers, GetOptions; and ACS optional calling methods: GetRPCMethods, RequestDownload, Kicked.
HAVE_REQUEST_DOWNLOAD	globals.h	Enables / disables RequestDownload method.

Compiler Flag	Location	Usage
HAVE_RPC_METHODS	globals.h	Enables / disables GetRPCMethods method. If GetRPCMethods is enabled it will be called on initial connection of CPE. GetRPCMethods call returns list of methods that can be called on ACS by the means of CPE. If some optional ACS methods are not supported on ACS they will be automatically switched off on CPE.
HAVE_SCHEDULE_INFORM	globals.h	Enables / disables ScheduleInform method.
HAVE_UDP_ECHO	globals.h	Enables / disables UDP Echo diagnostics (TR-143 standard).
HAVE_UPLOAD_ DIAGNOSTICS	globals.h	Enables / disables UploadDiagnostics method. Does not concern Upload method.
HAVE_VOUCHERS_OPTIONS	globals.h	Enables / disables both SetVouchers and GetOptions methods. Vouchers are set in the CPE by an ACS. And Vouchers' data structure instructs a particular CPE to enable or disable Options, and characteristics that determine under what conditions the Options persist.
HAVE_WANDSL_ DIAGNOSTICS	globals.h	Enables / disables WAN DSL diagnostics.
NO_LOCAL_REBOOT_ON_ CHANGE	globals.h	Disable automatic reboot when changes are made to parameters that normally require a reboot. This flag enables / disables ignoring reboot flag in the data-model.xml file (if local flag is enabled in data-model. xml file the reboot must be done by the ACS on parameter change). If NO_LOCAL_REBOOT_ON_CHANGE flag is enabled local flag is ignored and therefore a status return code of 1 is returned to signal that the parameter change has be done but not confirmed yet.
TR_111_DEVICE	Makefile	Defines that the client will operate as a TR-111 Device. This results in the client using an object model starting with "Device."
TR_111_DEVICE_WITHOUT_ GATEWAY	Makefile	Same as TR_111_DEVICE, however at startup the client will not wait for data for the Gateway to be filled before contacting the ACS.
TR_111_GATEWAY	Makefile	Defines that the client will operate as a TR-111 Gateway device. This results in the client using an object model starting with "InternetGatewayDevice."

Compiler Flag	Location	Usage
WITH_COOKIES	Makefile	This is a required flag for the Dimark Client. This flag should always be present. User has a possibility to compile Dimark client without cookies but the client will not be able to connect to Dimark ACS.
WITH_OPENSSL	Makefile	Required if you build with OpenSSL. This flag should always be present.
WITH_SOAPDEFS_H	globals.h	Enables / disables soapdefs.h file. This flag should always be present.
WITH_SSLAUTH	Makefile	Enable SSL Authentication through certificates.
WITH_STUN_CLIENT	Makefile	Enable STUN ACS connections
WITHOUT_DEBUG=TRUE	Makefile	Defines that compilation of client debug information will be disabled and it will not be included to code.

Building the Client on Red Hat Enterprise Linux

This section describes how to build the client code on a Linux system. While this section uses Red Hat Enterprise Linux distribution in the examples, it will work on all Linux distributions as well.

Required Additional Components

In order to build the client it is required that you have a soapcpp2 for your build platform.
This means that you have to obtain gSOAP (gSOAP Version 2.7.6c is required).

You could either use soapcpp2 binary file from http://sourceforge.net/projects/gsoap2/files/ (gSOAP Version 2.7.6c) or use patched gSOAP sources from http://www.dimark.com/client_release/index.html and compile binaries for your platform by yourself.

When completed, copy the binary gsoap_2.7.6c/soapcpp2 to the gsoap directory (.e.g `<dimark_client>/src/gsoap/` under the Dimark Client source directory.

Please consult ChangeLog.v4.0.txt and README Client files to check if any changes has been made concerning required gSOAP version.

It is also assumed that OpenSSL is already installed on your platform. If not, obtain any version of OpenSSL from 0.9.7f through 0.9.8i and follow the directions for compiling and installing.

Extract the Client

Create the user/tr069 directory and extract the client cod into it. Execute the following command to extract the client code:

```
tar xf <path to>/DPSClient-<Version>.tgz
```

This will extract the client source code into user/tr069 folder. You will as well need the provided gSOAP distribution in this place.

Perform the following command:

```
tar xf <path to>/DPSgSOAP-<Version>.tgz
```

to extract the gSOAP components that are needed to build the client.

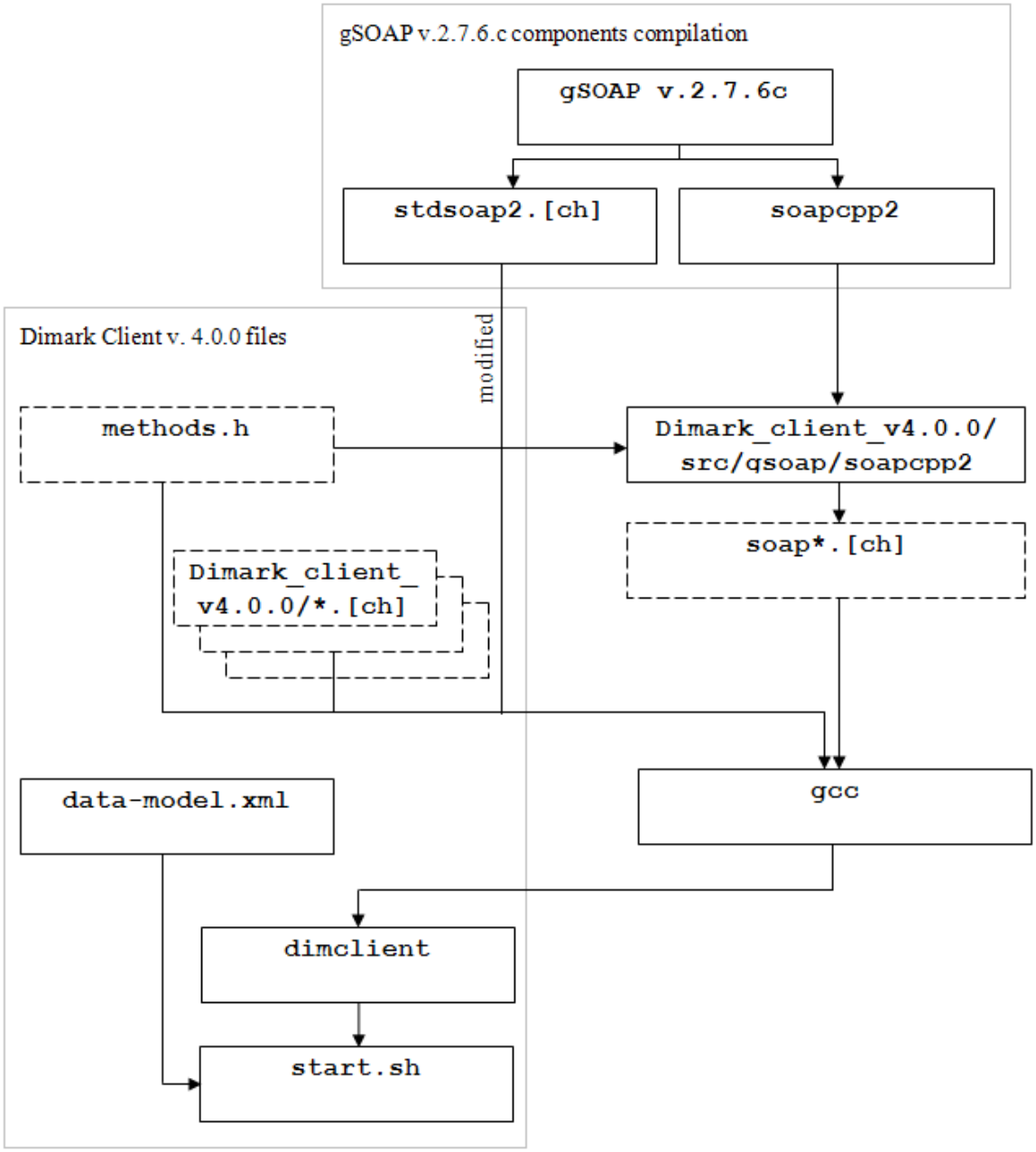
Once this is done copy the soapcpp2 program that was created in the previous step (required additional components) into the gsoap directory of user/tr069. Do not modify any other file that is in the gsoap directory or in the plugin directory. Those files are specifically modified to support TR-069 and replacing them with the original gSOAP files will make the client unusable.

Add the Client to the Build Process

Go to the Dimark client source directory. Review the Makefile and edit compiler flags and customize other settings. When ready to compile, type: `make`
The Dimark client will first call the soapcpp2 process to compile the gSoap directives to produce the necessary files to continue compilation. Some warning messages will be produced by soapcpp2. These are normal and can be ignored. The compilation will continue. When complete the following binary file will be present: dimclient
NOTE: See the start.sh script for how the client needs to be started and its target data directory prepared.
Start start.sh

You may need other compilation commands:
`make all` or `make` – project compilation;
`make release` – release preparation;
`make clean` – deletion of the intermediate files created during compilation and execution;
`make install` – project installation.

Compilation Process Scheme



6. Integration Process

The integration process consists of four steps:

- Expanding the TR-069 object model
- Implementing Init/Get/Set Methods
- Implementing Host interface
- Implementation of File System I/O (if required)

It is recommended to start by expanding the TR-069 Object model and then implementing the Set/Get functionality to retrieve and store parameter data from the host system. The final step should be adding the host interface to call the TR-069 client so that Notifications from the CPE to the ACS are possible.

Expanding the TR-069 Object Model (data-model.xml)

The client reads the data-model.xml configuration file (path to xml file is customizable in start.sh) and will be read on each startup of the client.

Based on cwmp-datamodel-1-2.xsd [14] Dimark generated extended data-model.xml file that is supported by Client. Developed xml schema allows to describe both basic datamodel and datamodel converted from dps.param (formerly used Dimark specific configuration file). To modify Broadband Forum xml schema (cwmp-datamodel-1-2.xsd) Dimark added new parameter attributes such as `notification`, `maxNotification`, `reboot`, `initIdx`, `getIdx` and `setIdx`.

If you have dps.param instead of xml configuration file, run an executable conv.jar file in console mode. It allows to convert dsp.param to xml file.

Sample command:

```
java -jar conv.jar dps.param data-model
```

where:

`dps.param` – parameter that allows to state name and path to xml file that will be created. If no path is stated (as in the sample) file will be created in the directory where conv.jar is located. Model described in xml will have the same name as xml file;

`data-model` – parameter that allows to state name and path to xml file that will be created. If no path is stated (as in the sample) file will be created in the directory where conv.jar is located. Model described in xml will have the same name as xml file.

XML Configuration File Description

The name of *.xml file is presented in the following form:

```
<model name="data-model">
```

As the result you will get data-model.xml configuratin file.

data-model.xml that is shipped in one package with Dimark Client describes what Object Model the client is using. Provided data-model.xml file includes the general DeviceInfo and ManagementServer settings. These are read-only data that lists CPE parameters and provides URL to contact the ACS.

1. TR-069 Parameter Name

To state the fully qualified TR-069 name of the parameter in data-model.xml you should use the following structure:

```
<object base="Device." access="readOnly" minEntries="1"
 maxEntries="unbounded">
  <object base="Device.ManagementServer." access="readOnly"
   minEntries="1" maxEntries="unbounded">
    <parameter base="URL" access="readOnly" notification="0"
     maxNotification="2" instance="0" reboot="1" initIdx="1"
     getIdx="1" setIdx="1">
    </parameter>
  </object>
</object>
```

where:

`Device.` – the top-level object for Device.ManagementServer TR-069 parameter;

`ManagementServer.` – the second-level object for Device.ManagementServer TR-069 parameter;

`URL` – the last element of Device.ManagementServer TR-069 parameter

`notification="0" maxNotification="2" reboot="1" initIdx="1" getIdx="1" setIdx="1"` – details that characterise the Device.ManagementServer TR-069 parameter (see their descriptions in the table below);

`access="readOnly" minEntries="1" maxEntries="unbounded"` – additional parameters that always should be included in every tag that defines object;

`access="readOnly" instance="0"` – additional parameters that always should be included in every tag that defines parameter.

2. Data Type

The data type of the element is represented by <syntax> tag in the following way:

```
<object ....>
    <parameter ... >
       <syntax>
         <string/>
       </syntax>
    </parameter>
</object>
```

where:

`string` – data type of the element.

There can be another types of the elements instead of `string`.

List of possible data types of the element (formerly presented as the second parameter in the dps.param syntax line):

- String (formerly 6 in the dps.param file);
- Integer (formerly 7 in the dps.param file);
- UnsignedInteger (formerly 9 in the dps.param file);
- Boolean (formerly 18 in the dps.param file);
- DateTime (formerly 11 in the dps.param file);
- Base64 (formerly 12 in the dps.param file);
- hexBinary;
- Int.

Types for DefaultObject (formerly 103, 104, 106, 115, 108, 109 in the dps.param file) are represented as corresponding data types for `default type="object"`.

3. Access List

Access List – comma separated list of access rights (i.e. Subscriber).

The following example of access rights has two entries – `x` and `y`:

```
<accessList>
  <entry>x</entry>
  <entry>y</entry>
</accessList>
```

If there is only one entry, the example will be as follows:

```
<accessList>
  <entry>x</entry>
</accessList>
```

4. Default Value

Default Value - the initial value of the parameter (only needed if Set/Get methods are not used).

Default Value is stated in the <syntax> tag below the data type tag and is presented as follows:

```
<syntax>
 <string/>
   <default type="factory" value="http://test.dimark.com:8080/dps/TR069"/>
</syntax>
```

5. Details that Characterise the TR-069 Parameter

An example from a TR-111 Device is:

```
<object base="Device." access="readOnly" minEntries="1"
maxEntries="unbounded">
   <object base="Device.ManagementServer." access="readOnly"
    minEntries="1" maxEntries="unbounded">
    <parameter base="URL" access="readOnly" notification="0"
     maxNotification="2" instance="0" reboot="1" initIdx="1"
     getIdx="1" setIdx="1">
    </parameter>
   </object>
</object>
```

where:

`notification="0" maxNotification="2" reboot="1" initIdx="1" getIdx="1" setIdx="1"` – details that characterise the Device.ManagementServer TR-069 parameter.

The following table represents the definitions of Details that Characterise the TR-069 Parameter:

Detail Element	Description
notification	Default setting for notification towards the ACS. 0 = no notification 1 = passive notification (during the next regular Inform) 2 = active notification (initiate an Inform when value changes) 3 = always include in Inform
maxNotification	The maximum permissible notification setting for this parameter (allows disabling of notifications through the ACS)
reboot	Flag to indicate if a Reboot is required if the value of the parameter is changed (from the ACS). 0 = Reboot 1 = No reboot required
initIdx	Allows calling an initialization function when the data is loaded from configuration file data-model.xml. Set to -1 if no initialization is required.
getIdx	The index number that was given to this parameter in the getArray[] of paramaccess.c Set to -1 if the parameter does not have a Get method, and is Write Only (i.e. Passwords). An entry of 0 reads the value from RAM An entry of 1 reads the data from the persistent layer
setIdx	The index number that was given to this parameter in the setArray[] of paramaccess.c. An entry of -1 treats the parameter as Read Only An entry of 0 causes the value to be read from the CPE An entry of 1 reads the data from the persistent layer

Creating Multiple Objects

TR-069 allows definition of Objects that appear multiple times under a single parent object.

The following example illustrates how nested objects can be created.

```
<object base="Device.Services.STBService.1.Capabilities.VideoDecoder."
 access="readOnly" minEntries="1" maxEntries="unbounded">
    <parameter base="VideoStandards" access="readOnly" notification="0"
     maxNotification="2" instance="0" reboot="0" initIdx="1" getIdx="1"
     setIdx="-1">
       <syntax>
         <string/>
         <default type="factory" value="MPEG2-Part2,MPEG4-Part4,MPEG4-
          Part10"/>
```

```
        </syntax>
      </parameter>
</object>
<object base="Device.Services.STBService.1.Capabilities.VideoDecoder.
 MPEG2Part2." access="readOnly" minEntries="1" maxEntries="unbounded">
    <parameter base="AudioStandards" access="readOnly" notification="0"
     maxNotification="2" instance="0" reboot="0" initIdx="1" getIdx="1"
     setIdx="-1">
       <syntax>
         <string/>
         <default type="factory" value="MPEG2-Part3-Layer2,MPEG2-Part3-
          Layer3"/>
       </syntax>
    </parameter>
    <parameter base="ProfileLevelNumberOfEntries" access="readOnly"
     notification="0" maxNotification="2" instance="0" reboot="0"
     initIdx="1" getIdx="1" setIdx="-1">
       <syntax>
       <unsignedInt/>
         <default type="factory" value="1"/>
       </syntax>
    </parameter>
</object>
<object base="Device.Services.STBService.1.Capabilities.VideoDecoder.
 MPEG2Part2.ProfileLevel." access="readOnly" minEntries="1"
 maxEntries="unbounded"/>
<object base="Device.Services.STBService.1.Capabilities.VideoDecoder.
 MPEG2Part2.ProfileLevel.1." access="readOnly" minEntries="1"
 maxEntries="unbounded">
    <parameter base="Profile" access="readOnly" notification="0"
     maxNotification="2" instance="0" reboot="0" initIdx="1" getIdx="1"
     setIdx="-1">
       <syntax>
         <string/>
         <default type="factory" value="HP"/>
       </syntax>
    </parameter>
```

```
    <parameter base="Level" access="readOnly" notification="0"
     maxNotification="2" instance="0" reboot="0" initIdx="1" getIdx="1"
     setIdx="-1">
       <syntax>
         <string/>
         <default type="factory" value="H-14"/>
       </syntax>
    </parameter>
    <parameter base="MaximumDecodingCapability" access="readOnly"
     notification="0" maxNotification="2" instance="0" reboot="0"
     initIdx="1" getIdx="1" setIdx="-1">
       <syntax>
         <string/>
         <default type="factory" value="10240"/>
       </syntax>
    </parameter>
</object>
```

Implementing a Set/Get Method

In order to implement an Init, Set or Get method it must be declared in the paramaccess.c module in the getArray[] or setArray[] respectively. These arrays contain a mapping from the index to the method name. There are many examples of access methods in the paramaccess.c file to use as a baseline for your own methods.

Once the method is implemented, the configuration file needs to be modified to include the Set/Get index number in the corresponding parameter entry.

Some of the index entries are already assigned by the client and cannot be changed.

- -1 No action;
- 0 Read/Write of the data is done without accessing the paramacess.c functionality;
- 1 Read/Write is done through paramaccess.c functionality.

Using entry 1 allows the client to handle the parameter in a read/write fashion without providing any backing implementation of the parameter. This way the client configuration file (data-model.xml) can be exploded to the full object model without completing the full integration work.

When defining index numbers, it is recommended to use the same number for both Set and Get methods to simplify debugging. Also, the client will use the Get index number during access from the host system when there is no Set access allowed from the access (ACS read-only parameter).

Implementing the Host Interface

Since TR-069 requires host system to communicate with the TR-069 client it provides not only the ACS interface, but also a secondary interface that allows the host system to retrieve and set client information.

This secondary interface uses an HTTP port (defined in ./src/globals.h that defaults to port 8081) that is used to communicate with the host system. This listener is started as a separate thread inside the client so that communication is possible even while the client is already communicating with the ACS.

The distinction of this system is that it does not have the same restrictions as the ACS (access to Read-Only parameter data). It allows access to any data stored inside the client.

In order to retrieve information from the Dimark TR-069 client GET and PUT methods had been implemented.

In the following examples the notation <NL> (LF, 0x0a) means the single newline character and not CRLF (0x0d 0x0a) as on Windows.

Retrieving Data from the Client

In order to read data from the client the following command would be issued over the host interface:

```
get<NL>
Parameterpath<NL>
```

The client would respond to this request with the following response:

```
Type<NL>
Parametervalue<NL>
Error code<NL>
```

This allows the CPE device to retrieve values that are stored inside the client environment. The code for this command is in hosthandler.c. The code calling the client must be done as part of the integration for parameters with notification support. This is done by issuing a HTTP GET from the host system towards the Dimark Client.

Setting Client Data

This command is important for the implementation as it can trigger notifications (or alarms) towards the ACS. It does not store the value if it is managed outside the client through Set/Get methods.

The structure of the call is:

```
set<NL>
Parameterpath<NL>
```

```
Parametervalue<NL>
```

A notification is sent to the ACS if necessary. The parametervalue is restricted to 1024 chars.

Adding Instance for an Object

In order to add new instance for an object the following command would be issued over the host interface:

```
add<NL>
Objectpath<NL>
```

The instancenumber of the new created object is returned, or if an error occurred a 0 is returned.

Deleting an Instance Object

In order to delete an instance object this method can be used. The format is:

```
del<NL>
Objectpath<NL>
```

If no error occurred a OK is returned.

Deleting an Option

This method can be used in order to delete an option which has timed out. The format is:

```
opt<NL>
VoucherSN<NL>
remove NL
```

Note: Some commands are available only for concrete data models if conditions (see table with available Compiler Flags in Compiler Flags for the Client section) are specified in code.

In this case vouchers options should be available:

```
#ifdef HAVE_VOUCHERS_OPTIONS
```

Retrieving Vendor Data

In order to read vendor data from the client the following command would be issued over the host interface:

```
GETVENDORINFO
 send vg
```

The client would respond to this request with the following response:

```
receive <errorcode> 0 = noError
receive <manufactorerOUI>
receive <serialNumber>
receive <productClass>
receive <emptyline>
```

Setting Vendor Data

In order to set vendor data the following command would be issued over the host interface:

```
SETVENDORINFO
 send vs
 send <manufactorerOUI>
 send <serialNumber>
 send <productClass>
```

The client would respond to this request with the following response:

```
receive <errorcode> 0 = noError
receive <emptyline>
```

Adding Device Data

In order to add data about connecting device the following command would be issued over the host interface:

```
ADDDEVICEINFO
 send da
 send <manufactorerOUI>
 send <serialNumber>
 send <productClass>
```

The client would respond to this request with the following response:

```
receive <errorcode> 0 = noError
receive <emptyline>
```

Deleting Device Data

In order to delete device data the following command would be issued over the host interface:

```
REMOVEDEVICEINFO
 send dr
 send <manufactorerOUI>
 send <serialNumber>
 send <productClass>
```

The client would respond to this request with the following response:

```
 receive <errorcode> 0 = noError
 receive <emptyline>
```

Retrieving ACS URL

In order to read ACS URL the following command would be issued over the host interface:

```
DHCPDISCOVERY
 send hd
 send <manufactorerOUI>
 send <serialNumber>
 send <productClass>
```

The client would respond to this request with the following response:

```
 receive ho
 receive <manufactorerOUI>
 receive <serialNumber>
 receive <productClass>
 receive <URL>
 receive <emptyline>
```

Retrieving Gateway Data

In order to connect to Gateway device sends request with its own parameters and receives Gateway parameters to establish a connection. The format is:

```
DHCPREQUEST
 send gr
 send <manufactorerOUI>
 send <serialNumber>
 send <productClass>
```

The client would respond to this request with the following response:

```
receive ho
receive <manufactorerOUI>
receive <serialNumber>
receive <productClass>
```

7. Additional Parameters

Additional parameters (specific for Dimark Client and not listed in any standards) were added to initial parameter file ().

`Dimark.` object contains all additional Dimark Client specific parameters.

Previously all transfers were allowed only from/to ACS. `TransferURL` parameter allows to designated location from which downloads/uploads will be started.

For Device.Dimark:

```
<object base="Device.Dimark." access="readOnly" minEntries="1"
maxEntries="unbounded">
    <parameter base="TransferURL" access="readOnly" notification="0"
     maxNotification="2" instance="0" reboot="1" initIdx="-1"
     getIdx="0" setIdx="-1">
       <syntax>
         <string />
         <default type="factory" value="[IP|URL]" />
       </syntax>
    </parameter>
</object>
```

where:

`IP|URL` is an IP address or URL of the site from which Download/Upload methods are allowed to be invoked besides ACS (for Download/Upload methods call format see section 15. Download and Upload Methods Support).

For InternetGatewayDevice.Dimark:

```
<object base="InternetGatewayDevice.Dimark." access="readOnly"
 minEntries="1" maxEntries="unbounded">
    <parameter base="TransferURL" access="readOnly" notification="0"
     maxNotification="2" instance="0" reboot="1" initIdx="-1"
     getIdx="0" setIdx="-1">
       <syntax>
         <string />
```

```
          <default type="factory" value="[IP|URL]" />
        </syntax>
    </parameter>
</object>
```

where:

`IP|URL` is an IP address or URL of the site from which Download/Upload methods are allowed to be invoked besides ACS (for Download/Upload methods call format see section 15. Download and Upload Methods Support).

8. Implementing Parameter Storage

The storage interface used by the client has been completely redesigned in release 3.0 and improved in further versions of the Dimark TR-069 Client.

In previous versions of the Dimark TR-069 Client, data was saved with both the current settings as well as Meta information (i.e. Access rights). Now those information sections are saved separately. The complete storage interface has been extracted into the file host/parameterStore.c. By separating those two elements, flash storage requirements are decreased since only persistent and changeable parameters are saved.

The supplied implementation counts on saving the data in two distinct directories. In each case the parameter name will be used as the filename as well.

The following functionality must be implemented in parameterStore.c. to read the initial parameter file (data-model.xml) line by line and use the callback function to perform additional work on each of the parameters and store the data:

```
int loadInitialParameterFile(newParam *callbackF)
```

In case the parameter defines an Init method it will call this Init method for the parameter.

The following function loads a single parameter file and create the parameter in memory:

```
int loadSingleParameterFile( const char *, newParam *callbackF )
```

The following adds a new parameter and saves the metadata (notification, access rights) of the parameter in persistent memory:

```
int storeParameter(const char *name, const char *data)
```

The metadata is given as part of the string and are identical to the entries in data-model.xml. The name is the complete fully qualified parameter name. The value of the parameter is not provided.

The following function is called when metadata of a parameter needs to be updated. The name field is the fully qualified name of the parameter:

```
int updateParameter(const char * name, const char *data)
```

The following function removes a parameter from persistent memory. This needs to remove both the metadata

as well as the parameter value information. This function is called during Delete Object functionality:

```
int removeParameter(const char *name)
```

The following function removes all parameter information. This function is called during a FactoryReset() that is triggered by the ACS:

```
int removeAllParameters(void)
```

The following function saves the value of a parameter given as name in the first argument. Since the value is dependent of the type, the type is given as well with the parameters:

```
int storeParamValue(cont char *, ParameterType, ParameterValue *)
```

The following function retrieves the value of a parameter, which name is in first argument:

```
int retrieveParamValue(const char *, ParameterType, ParameterValue *)
```

The following function needs to be implemented for checking of a parameter data and/or metadata file existence:

```
unsigned int checkParameter(const char *)
```

The return value of this function can be one of the following:

- 0 File does not exist;
- 1 Data file exists;
- 2 Metadata file exists;
- 3 Both Data and Metadata files exist.

9. Implementing Storage Environments

Storing of events, file transfers and options are done similar to the storage of parameters. In case the provided implementation needs to be changed, consult the host directory (host/eventStore.c., host/filetransferStore.c. and host/optionStore.c. files) for a sample implementation.

Events

The following function creates the storage environment for events. This typically results in a single file being created where events can be stored in the file system. Events are defined in TR-069 (i.e. “0 BOOTSTRAP”, “8 DIAGNOSTICS COMPLETE”, “2 PERIODIC”, etc):

```
int createEventStorage(void)
```

The following function reads all events from the storage (created with createEventStorage) and calls newEvent() to process it in dimclient:

```
int readEvents(newEvent *func)
```

The following function removes all events from the event storage (created with createEventStorage):

```
int clearEventStorage(void)
```

The following function insert the event string (data) at the end of event storage (created with createEventStorage):

```
int insertEvent(const char *data)
```

The following bootstrap marker allows the client to detect if it’s in an “initial” state (after factory reset or first time boot) or if this is a regular reboot situation. In the default implementation there is a marker inside the file. This marker must be retained over reboot events:

```
int createBootstrapMarker(void)
```

When the ACS issues a Factory Reset, the following method will be called to remove the Bootstrap marker from the event list and allow the client to appear with a BOOTSTRAP event against the ACS:

```
int deleteBootstrapMarker(void)
```

The following function is called during startup of the client to identify if the bootstrap marker is present. Returns true if bootstrapMarker is found else returns false:

```
bool isBoostrapMaker(void)
```

File Transfers

The following function reads the list of open (pending) file transfers at the boot of the client. For each line the callback function has to be called:

```
int readFtInfor(newFtInfo *callbackF)
```

The following function saves the file transfer information with a specified name. The name is unique for each transfer:

```
int storeFtInfo(const char *name, const char *data)
```

The following function deletes the transfer data associated with a given name:

```
int deleteFtInfo(const char *name)
```

The following function deletes all informations about pending file transfers:

```
int clearAllFtInfo(void)
```

The following function returns a default filename as destination for a download in case the ACS did not deliver a destination filename in the download request:

```
const char * getDefaultDownloadFilename(void)
```

Options

The following function is for processing vouchers, which are containers for options. The voucher must first be saved in a file. This file is used by gSOAP to read the XML information of the voucher:

```
const char *getVoucherFilename(int index)
```

The following function reads all options and initiates callback function for each dataset:

```
int reloadOptions(newOption callback())
```

The following function deletes all option data records:

```
int deleteAllOptions(void)
```

The following function deletes information for a named option:

```
int deleteOption(const char *name)
```

The following function saves data record under the provided name argument. The name field is unique:

```
int storeOption(const char *name, const char *data)
```

10. Callbacks

Callbacks allow custom activities to be carried out in the client during specific situations. The defined callbacks are:

- Before the client makes a call to the ACS (preSessionCbList);
- After terminating the session with the ACS (postSessionCbList);
- Clean up when temporary storage is freed (cleanUpCbList).

The client needs to register the callback by calling the method addCallback, which is defined as:

```
void addCallback(Callback, List *);
```

The callback itself is defined as:

```
int (*Callback) (void);
```

No parameters are provided to those callbacks. The return value of the callback method itself can have the following values:

Return	Description
CALLBACK_REPEAT	Will call the same callback the next time the trigger situation happens.
CALLBACK_STOP	Removes the callback from the list. The callback function will not be called again.

Callback functions are called in the order they are registered.

11. Diagnostics Support

Dimark Client supports following TR-143 diagnostics:

Download Diagnostics utilizing FTP transport and Download Diagnostics utilizing HTTP transport [15, 16].
The Download Diagnostics test is being used to test the streaming capabilities and responses of the CPE and the WAN connection.

Upload Diagnostics utilizing FTP transport and Upload Diagnostics utilizing HTTP transport [15, 16].
The Upload Diagnostics test is being used to test the streaming capabilities and responses of the CPE and the WAN connection.

UDP Echo is being used for monitoring. CPE acts as server and listens for UDP datagrams on UDP port 8088. When a datagram is received, the data from it is sent back in an answering datagram.

For further TR-143 diagnostics details please see TR-143, Enabling Network Throughput Performance Tests and Statistical Monitoring, Broadband Forum Technical Report.

You can browse skeleton function for IP Ping Diagnostics in `src/host/diagParameter.c`.

12. Kicked RPC Support

Kicked RPC is supported by the Dimark Client v.3.0.10 and higher. Detailed description can be found in TR-069 CPE WAN Management Protocol v1.1 Amendment 2 (Annex A 4.2.1 and Annex D).

Sample of Kicked procedure invocation:
Type `http://<CPE_IP>:8084/command=com&arg=arg&next=NextURL` in your browser.
Where 8084 is port which CPE is listening to be kicked.
CPE has a DoS attaks protection mechanism.
Procedure invocation is case sensitive. Be sure you specify all parameters as they are shown in sample.

If Kicked RPC has been initiated successfully, browser is redirected to
`http://<CPE_IP>:8084/NextURL` page, defined in ACS KickedResponse RPC.

Otherwise you are redirected to `http://<CPE_IP>:8084/FaultURL` page or get HTTP 503 Error that mean that DoS attaks protection was not passed or site URL does not coincide with Device.ManagementServer. KickURL or InternetGatewayDevice.ManagementServer. KickURL.

13. Download and Upload Methods Support

Download and Upload methods are supported by the Dimark Client. Dimark Client can transfer specified file from/to designated location. Detailed description can be found in TR-069 CPE WAN Management Protocol v1.1 Amendment 2 (Annex A 3.2.8 and Annex A 4.1.5).

Sample of Download method invocation:

```
http://192.168.15.17:8096/command=Download&type=1 Firmware Upgrade Image&url=http://www.insart.com/images/stories/content/homepage_logo.jpg&username=user&password=name&delay=10000&size=24024&target=target&success=success&failure=failure
```

Sample of Upload method invocation:

```
http://192.168.15.17:8096/command=Upload&type=2 Vendor Log File&url=http://www.insart.ua/post&username=user&password=name&delay=10
```

Here 8096 is port which CPE is listening to start downloads/uploads.

CPE has a DoS attacks protection mechanism.

Browser will generate corresponding success or failure message:

- HTTP 503 Error – transfer was not initiated as either DoS attacks protection was not passed or `command` parameter is incorrect or not specified or `url` parameter is not specified;
- transfer not accepted – transfer initiation was failed;
- transfer accepted – transfer was initiated (but that doesn't mean it have been started as delay time can be set).

Note: Only `command` and `url` are required parameters. If values of optional parameters are unknown they shouldn't be mentioned in methods invocation.
Methods invocation is case sensitive. Be sure you specify all method parameters as they are shown in samples. Sequence order of required parameters doesn't matter and their position can be shifted.

Sample of Download method invocation (required parameters):

```
http://192.168.15.17:8096/command=Download&url=http://www.insart.com
```

14. Starting Several Clients

Several Clients can be started on one machine (using one IP). In start.sh file user changes default base 8080 port to any port not intersecting ports that are already in use. After that related ports are set automatically (each of them is formed according to following incrementation pattern: ACS port = default base 8080 port +2, host port = default base 8080 port +1, kick port = default base 8080 port +4, UDP Echo port = default base 8080 port +8).

Note: Default base 8080 port is used if in start.sh file following line can be found:
`until ./dimclient`

For example, if instead of `until ./dimclient` user specifies `until ./dimclient 8090` then ACS port will be 8092, host port – 8091, kick port – 8094, file transfer (download/upload) port – 8096, UDP Echo port – 8098. While starting next Client(s) remember those ports are occupied and cannot be used as base port value in start.sh file.

For given example you cannot use any of the following ports as base port for next starting Client: 8090, 8091, 8092, 8094, 8096, 8098.

Please remember that other ports can be occupied for STUN server or other needs. Check data-model.xml file to be sure port you want to specify as next Client's base port is available.

Note: If UDP Echo port is set in data-model.xml explicitly set port will be used as default instead of one set using incrementation pattern.

15. Log Configuration

Dimark Client log can be configurated using log.config file located in root directory.

The following levels of logging exist in Dimark Client:

- DEBUG – debug data is logged (loggs everything). DEBUG logging level is implemented for environments where no debugging functionality is available. This option allows generating output of the client to help diagnose issues. Debug functionality is defined in debug.h and implemented in debug.c;
- INFO – information is logged;
- WARN – warnings are logged;
- ERROR – logs errors concerning stated subsystem.

Note: DEBUG level is the highest and loggs all data including INFO, WARN and ERROR. Each subsequent level include all logging levels that are below it.
If there are no log.config file, the default levels of subsystem logging will be INFO.

Debugs can be removed completely from the output and the program code. This is managed through compiler flag `WITHOUT_DEBUG=TRUE` in Makefile.

Each subsystem has assigned logging level.

The syntax of log.config file is as follows:

```
subsystem;level
```

The following sample includes all available subsystems and shows the example of log.config syntax:

```
SOAP;DEBUG <NL>
MAIN;INFO <NL>
PARAMETER;WARN <NL>
TRANSFER;ERROR <NL>
STUN;DEBUG <NL>
ACCESS;DEBUG <NL>
MEMORY;DEBUG <NL>
EVENTCODE;DEBUG <NL>
SCHEDULE;DEBUG <NL>
ACS;DEBUG <NL>
VOUCHERS;DEBUG <NL>
OPTIONS;DEBUG <NL>
```

```
DIAGNOSTIC;DEBUG <NL>
VOIP;DEBUG <NL>
KICK;DEBUG <NL>
CALLBACK;DEBUG <NL>
HOST;DEBUG <NL>
REQUEST;DEBUG <NL>
DEBUG;DEBUG <NL>
AUTH;DEBUG <NL>
<EOF>
```

where:

<NL> – new line symbol;

<EOF> – end of line symbol.

16. Client Error Codes

The following table describes the Dimark specific error codes that can be generated. These error codes are specific to the Dimark Client and extend the TR-069 Error code table. Developers who will extend the Client and include additional error codes must make sure that they are using different error codes.

Code	Constant	Description
9800	ERR_DIM_EVENT_WRITE	Unable to open persistent file for writing. This results in the client being unable to communicate status after reboot events.
9801	ERR_DIM_EVENT_READ	Unable to open persistent file for reading. This results in the client being unable to communicate status information to the ACS.
9810	ERR_DIM_TRANSFERLIST_WRITE	Unable to write a file transfer entry to persistent storage. This will result in the client being unable to process a DownloadRequest.
9811	ERR_DIM_TRANSFERLIST_READ	Unable to read file transfer entries from persistent storage. This results in the client being unable to process a DownloadRequest and its status to the ACS.
9820	ERR_INVALID_OPTION_NAME	The provided option does not exist in the CPE. This fault is only generated on the HOST interface and not used with the ACS.
9821	ERR_CANT_DELETE_OPTION	Unable to Delete a Voucher Entry This fault is only generated towards the HOST interface and not used with the ACS.
9822	ERR_READ_OPTION	Unable to read voucher information from persistent storage.
9903	ERR_WRITE_OPTION	Unable to write voucher information to persistent storage.

References

1. TR-069, CPE WAN Management Protocol (CWMP), Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-069_Amendment-2.pdf.
2. TR-098, Internet Gateway Device Data Model for TR-069, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-098_Amendment-2.pdf.
3. TR-104, DSLHomeTM Provisioning Parameters for VoIP CPE, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-104.pdf.
4. TR-110, DSLHomeTMApplying TR-069 to Remote Management of Home Networking Devices, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-110v1.01.pdf.
5. TR-106, Data Model Template for TR-069-Enabled Devices, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-106_Amendment-4.pdf.
6. TR-111, DSLHomeTMApplying TR-069 to Remote Management of Home Networking Devices, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-111.pdf.
7. TR-135, Data Model for TR-069 Enabled STB, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-135.pdf.
8. TR-140, TR-069 Data Model for Storage Service Enabled Devices, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-140_Amendment-1.pdf.
9. TR-142, Framework for TR-069 enabled PON Devices, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-142_Issue-2.pdf.
10. TR-143, Enabling Network Throughput Performance Tests and Statistical Monitoring, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-143_Corrigendum-1.pdf.
11. TR-157, Component Objects for CWMP, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-157_Amendment-3.pdf.
12. TR-181, Device Data Model for TR-069, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-181_Issue-1.pdf, http://www.broadband-forum.org/technical/download/TR-181_Issue-2.pdf.
13. TR-196, Femto Access Point Service Data Model, Broadband Forum Technical Report, http://www.broadband-forum.org/technical/download/TR-196.pdf.
14. cwmp-datamodel-1-2.xsd, Broadband Forum XML Schema http://www.broadband-forum.org/cwmp/cwmp-datamodel-1-2.xsd
15. RFC 2616, Hypertext Transfer Protocol – HTTP/1.1, http://www.ietf.org/rfc/rfc2616.txt
16. RFC 2617, HTTP Authentication: Basic and Digest Access Authentication, http://www.ietf.org/rfc/rfc2617.txt

Annex A. TR-069 Client FAQ

Please see http://www.dimark.com/client_release/index.html for the latest information.

General Questions

Q: Client memory consumption has Threads parameter. What does it mean?

A: Dimark Client has two main threads – Main and Host. They cannot be turned off. Other possible threads are: Connection Request, UDP Echo, Transfers, Kick and STUN. Number reflecting the threads depends on compiler flags that are set.

Q: We will be targeting non-Linux platforms and will need to port to our platform OS, BSD socket interface, and SSL interface. We would like to understand what level of support we can expect from Dimark for this port/ integration effort.

A: Our focus is on deployments and it is in Dimark's best interest to get our partners shipping products ASAP. We support any questions via email on how to bring the client code to the target platform as well on how to integrate the client with the host system.

Q: Based on your experience with your client, what is the estimated port / integration time to a new platform (consider our target is not Linux based).

A: The typical porting for a Linux (or Linux look-alike, such as VxWorks) has been between 1 day and 1 week. If your platform is non Linux based, but provides socket interfaces. The porting work is primarily for SOAP.

Q: What about WinCE?

A: The client can run on Windows (gSOAP and OpenSSL are portable to those platforms). All that is required is a special library (called pthreads) to operate the client.

Q: Can you provide details of the test and certification services? The proposal states our devices will be fully certified in your Certification Lab. What if we would like additional passes, for example, for a new platform or for an extended data model?

A: The certification is based on a single CPE. Additional fees would arise for additional CPE systems. Multiple passes (within a reasonable limit) until passing the certification of a single CPE are expected.

Q: What are all the system resources required? (threads/tasks, sockets, timers, semaphores, etc.)

A: BSD sockets, threads. The client does not use semaphores or other means. There is another port used for communication with the host system.

Q: Are any tools required to build the sources beyond the basics GCC toolchain?

A: SSL is not part of our distribution of the client code. We provide the hooks for OpenSSL with our client. Additionally we are currently working on Mocana NanoSSL.

Q: Is there any third party copyrighted code provided or required beyond Dimark's? Any additional licenses, sub-licenses we would need? Any open source and/or GPL code? (beyond gSOAP)?

A: OpenSSL which can be replaced with Mocana NanoSSL in the near term. gSOAP is provided with the client. However gSOAP libraries are not linked with the client.

Q: We already have a gSOAP client and understand you can help us modify it for TR-069 compliance. Will these modifications disrupt our existing SOAP protocols?

A: If you already have gSOAP on the system, the majority of porting work for the client has already been done, significantly reducing the porting time for the client to your platform. The modifications for TR-069 will not affect other gSOAP operations. However we require a specific gSOAP version to be used with our client due to the modifications (Version 2.7.6c). The modification has to do with the bi-directional operation of TR-069, which is outside of the intended scope of SOAP. gSOAP is used as source code generator for the SOAP operations.

Q: What Linux Version is the client compatible with?

A: The client is compatible with any Linux Version (2.2, 2.4, 2.6) as well as almost any other operating system.

Engineering Questions

Q: What gSoap version do I need?

A: We require using gSOAP version 2.7.6c

Q: Should I overlay gsoap-2.7.6c directory files with the files from the Dimark_TR-069_Client_gSOAP package?

A: You use the gSOAP 2.7.6c package to build soapcpp2 which is called in the Makefile to build the client. Do not overlay any files from the client with those from gSOAP 2.7.6c.

Q: Do I need to rename (or symbolically link) the directory gsoap-linux-2.7 to gsoap.

A: See the above answer. Do not modify the contents of the gsoap directory of the client.

Q: When I use "make", "make clean" and then "make", I get an error message. Could Dimark tell me that if it is bug or not?

A: This is a known issue with make. For some obscure reason make does not recognize the creation of soapC.c after a "make clean" is done. Starting make a second time will cure the problem.

Q: I modify the Makefile to make the source code without SSL and I get some error message. Could Dimark tell me that's the problem?

A: The configuration does still include the DIGEST authentication, which in turn requires SSL libraries being present. Removing the plugin/httpda.c and plugin/smdevp.c modules will cure this situation.

Q: Could Dimark give me the sample configuration files for the program to access the testing server, "http://test.dimark.com:8080/login"?

A: Define http://test.dimark.com:8080/login as current InternetGatewayDevice.ManagementServer.URL in the data-model.xml. The client will take the OUI-SNR as Username and Password per TR-069. Our ACS will allow connectivity with those credentials.

Runtime Questions

Q. The client log shows ACS Notification failed message whenever there is connection request from ACS. Here is log for a connection request from ACS for TraceRouteDiagnostics. The Diagnostics runs properly and reports results back to ACS.

```
<==================== LOG ======================>
soap_serve endet 0 env:Envelope :
GetAccess: 1 Device.ManagementServer.PeriodicInformEnable 0x11fe50
GetAccess: 1 Device.ManagementServer.PeriodicInformInterval 0x11e4c8
ACS Accept ret: 13
GetAccess: 119 Device.ManagementServer.ConnectionRequestURL 0x11f000
GetAccess:    1    Device.ManagementServer.ConnectionRequestUsername
0x11ee80
ACS Notification failed: 401      <--------> Error Message
ACS Accept ret: 13
Digest:    username="dps"
Key: username Value: "dps
Digest:     realm="Dimark"
Key: realm Value: "Dimark
```

```
Digest: nonce="1234567890ABCDEF"
Key: nonce Value: "1234567890ABCDEF
Digest:    uri="/acscall"
Key: uri Value: "/acscall
```

A. This is normal behavior as the ACS will not include the Authentication data in the first transmission. This allows the client to send the DIGEST information to the ACS which in turn will resend the request with the DIGEST Authentication information. This is normal HTTP behavior.

Q. When running the client the following messages appear on the console:

```
plugin/httpda.c(192): free(0x1002dcb8) pointer not malloced
plugin/httpda.c(192): free(0x1002b378) pointer not malloced
plugin/httpda.c(192): free(0x1002f250) pointer not malloced
plugin/httpda.c(192): free(0x1002e7d8) pointer not malloced
plugin/httpda.c(192): free(0x1002e2e0) pointer not malloced
plugin/httpda.c(192): free(0x1002f760) pointer not malloced
```

A. This happens when the client is compiled with the SOAP_DEBUG flag. Unfortunately the gSOAP DIGEST authentication component does not fully adhere to the coding standard of gSOAP and as a result, this message is displayed. It does not indicate a memory leak.

Q: Why Basic authentication request is sent for Download regardless authorization type defined by ACS?

A: Download process for Dimark Client v 3.0.10 and higher has the following peculiarity – regardless using authorization type, when download is initiated Basic authorization is sent by default. If server rejects initiating download with Basic authorization, Digest authorization is sent.
Such behavior doesn't contradict HTTP protocol standards. Problems can occur only if server doesn't correspond to HTTP 1.0 standard.

TR-111

Q: Which open source STUN software should I use?

A: The implementation we have seen used the most is from Sourceforge:
http://sourceforge.net/projects/stun/
This software works well with both Dimark's ACS and client. Integration with the client is required.
Please note: Dimark doesn't include STUN software with our distribution as interoperability issues exist between different implementations. While the STUN software listed above is widely used, it is important to confirm interoperability between the STUN client and server in a production environment. Industry efforts are underway to correct the situation and provide a standard solution.