# TR-069 Implementation Notes

## For BCM963xx DSL Linux

### Version 3.04 with tr69c_3_05L_14 patch and beyond

16215 Alton Parkway • P.O. Box 57013 • Irvine, CA 92619-7013 USA • Phone: (949) 450-8700 • Fax: (949) 450-8710

# Table of Contents

## REVISION HISTORY

| *Revision Number* | *Date* | *Change Description* |
|---|---|---|
| V1.0 | 04/10/06 | Initial Release. |

## 1.0 INTRODUCTION

This document serves as an implementation notes on the details of the TR-069 WAN management application that is supported in Linux releases (3.04 with tr69c_3_05L_14 patch and beyond) of BCM963xx platforms. It is intended to be a great aid in showing user how to modify some standard TR-069 parameters, and add standard or vendor specific proprietary TR-069 objects to Broadcom TR69C application in binary format. This mechanism can only be applied to Linux 3.04 or 3.06 releases with tr69c_3_05L_14 patch and beyond.

It is expected that this feature will be desirable to customer, field and development engineering teams.

## 2.0 TR69C PARAMETERS FRAMEWORK

TR-069 parameters are defined in TR-069 specification at Appendix B. The tr69c CPE parameter framework uses a similar tree structure to represent the parameter supported by the BCM963xx reference design. These TR-069 parameters are linked together and organized as a tree structure with the root of the tree at “InternetGatewayDevice”.

### 2.1 TR69C Object Node

TR-069 parameter is represented by the following **TRxObjNode** C structure definition in tr69c/inc/tr69cdefs.h:

```
typedef struct TRxObjNode {
   const char  *name;
   TRxPAttrib  paramAttrib;
   TRxSETFUNC  setTRxParam;    /* only set if parameter is writeable */
   TRxGETFUNC  getTRxParam;
   void        *objDetail;
   InstanceDope *instanceDope;
} TRxObjNode;
```

Where **TRxPAttrib** is the following C structure

```
typedef union TRxPAttrib {
   struct Attrib {
   eTRxType    etype:8;
   unsigned    slength:16;
   unsigned    inhibitActiveNotify:1; /* set to always inhibit change notification: use on counters */
   } attrib;
   InstanceDesc    *instance;
} TRxPAttrib;
```

**TRxSETFUNC** and **TRxGETFUNC** are defined as following

```
typedef TRX_STATUS (*TRxSETFUNC)(const char *value);
typedef TRX_STATUS (*TRxGETFUNC)(char **value);
#define TRXGFUNC(XX) TRX_STATUS XX (char **)
#define TRXSFUNC(XX) TRX_STATUS XX (const char *)
```

and **InstanceDope** is the following C structure

```
typedef struct InstanceDope {
    struct InstanceDope *next;
    InstanceDesc *instance;
    char *pdata;
    unsigned notification:2;
    unsigned accessListIndex:1;
} InstanceDope;
```

## 2.2 TR69C Object Type

The **TrxObjNode** represents the TR-069 parameter in the tr69c tree. One of important field in this node is **paramAttrib** that has **etype** field to specify the type of this node. The following **eTRxType** C enum defines the type of parameter node in tr69c tree.

```
typedef enum {
    tUnknown=0,
    tObject,
    tString,
    tInt,
    tUnsigned,
    tBool,
    tDateTime,
    tBase64,
    tInstance,
    tStringSOnly
} eTRxType;
```

Where:

- tUnknown: unknown object type, and should not be used
- tObject: static partial path object node that does not have any specific data type. Ex: InternetGatewayDevice.ManagementServer.
- tString: leaf node that has its data type as string. Ex: InternetGatewayDevice.ManagementServer.URL
- tInt: leaf node that has its data type as integer. It's used for Status and Notification.
- tUnsigned: leaf node that has its data type as unsigned integer. Ex: InternetGatewayDevice.ManagementServer.PeriodicInformInterval
- tBool: leaf node that has its data type as Boolean. Ex: InternetGatewayDevice.ManagementServer.PeriodicInformEnable
- tDateTime: leaf node that has its data type as SOAP dateTime. Ex: InternetGatewayDevice.ManagementServer.PeriodicInformTime
- tBase64: leaf node that has its data type as Base64 encoded binary. Ex: InternetGatewayDevice.UserInterface.ISPLogo
- tInstance: dynamic partial path object node that does not have any specific data type. It should be used in AddObject and DeleteObject RPC methods. Ex: InternetGatewayDevice.Layer3Forwarding.Forwarding.
- tStringSOnly: leaf node that has its data type as string. It's used for parameters that allow Set only, but when Get, it only returns empty string. Ex: InternetGatewayDevice.ManagementServer.Password

## 2.3 TR69C Macros

### 2.3.1 SVAR(X) and SSVAR(X,Y)

The preprocessor macros SVAR(X) and SSVAR(X,Y) are defined in tr69c/inc/tr69cdefs.h. They have two versions. If CPEVARNAMEINSTANCE is defined, the SVAR(X) macro will define a character string constant(X) and its C label(X). If CPEVARNAMEINSTANCE is not defined, the SVAR(X) macro will define an extern to the string constant label. The macro is used in header files to define all of the parameter string constants. When included in C file, an instance for each string is created. Otherwise, an extern is defined for the string label.  In case where the parameter name is not a valid C label use the SSVAR(X,Y) which define string Y with label X. The use of this macro limits the number of instances of string constants in the various compiler objects.

### 2.3.2 TRxSETFUNC(XX) and TRxGETFUNC(XX)

The preprocessor macros TRxSETFUNC(XX) and TRxGETFUNC(XX) are defined in tr69c/inc/tr69cdefs.h. They should be used in C file to define get/add or set/delete handler functions that are associated with the full path (leaf) node parameter. TRxSETFUNC macro should be used to define set handler function or delete handler function, and TRxGETFUNC macro should be used to define get handler function or add handler function.

## 2.4 TR69C Get/Set handlers

The **TrxObjNode** also contains two important fields: setTRxParam, and getTRxParam that provide callback mechanism for setting or getting parameter value.

### 2.4.1 Get Handler Function

getTRxParam is a pointer to the handler function that will be called when tr69c receives GetParameterValues RPC method on this parameter.

The prototype for the get handler function is:

```
TRX_STATUS geFuncName(char **value);
```

The framework calls the function with a pointer to the value pointer. The getter functions copy the parameter value as a null terminated character string to an allocated buffer. The pointer to the buffer is written to the **value and the function return value is a TRX_STATUS value of TRX_OK, TRX_REBOOT, or TRX_ERR. The following is an example of the get of the Specification Version parameter:

```
TRX_STATUS getSpecVersion(char **value) {
        Value = strdup(“1”);
        return TRX_OK;
}
```

The framework will release the allocated memory when it has finished building the RPC response message. There is no restriction on multiple parameters using the same callback function.

### 2.4.2 Set Handler Function

setTRxParam is a pointer to the handler function that will be called when tr69c receives SetParameterValues RPC method on this parameter.

The prototype for the set handler function is:

```
TRX_STATUS setFuncName(const char *value);
```

The parameter value is passed as a null terminated character string. The string value is released upon return to the framework from the setter function. The following is an example of the set handler function for setting the .ManagementServer.ConnectionRequestPassword:

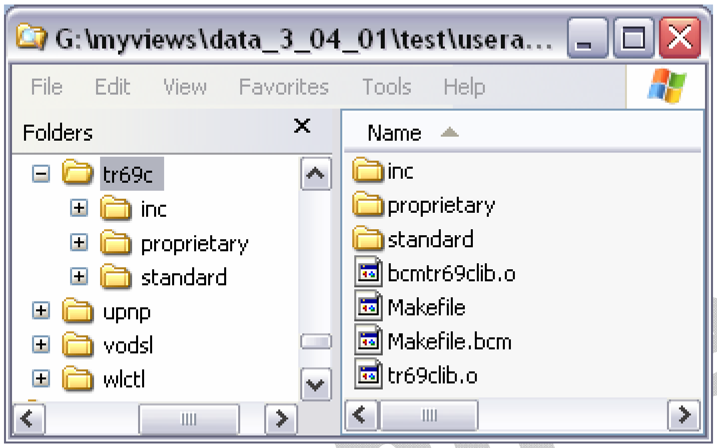
```
TRX_STATUS setConnectionPassword(const char *value) {
        setPassword(value);
        return TRX_OK;
}
```

# 3.0 TR69C FILE STRUCTURE

This section describes the file structure of tr69c sources that are released to customers in 3.04 or 3.06 releases with tr69c_3_05L_14 patch and beyond.

## 3.1 tr69c Directory

The file structure under tr69c directory should be as following where bcmtr69clib.o is Broadcom tr69c application in binary format, and tr69clib.o is the final tr69c application that includes vendor specific proprietaryTR-069 parameters in proprietary directory, some standard TR-069 parameters in standard directory, and parameters that are supported in bcmtr69clib.o

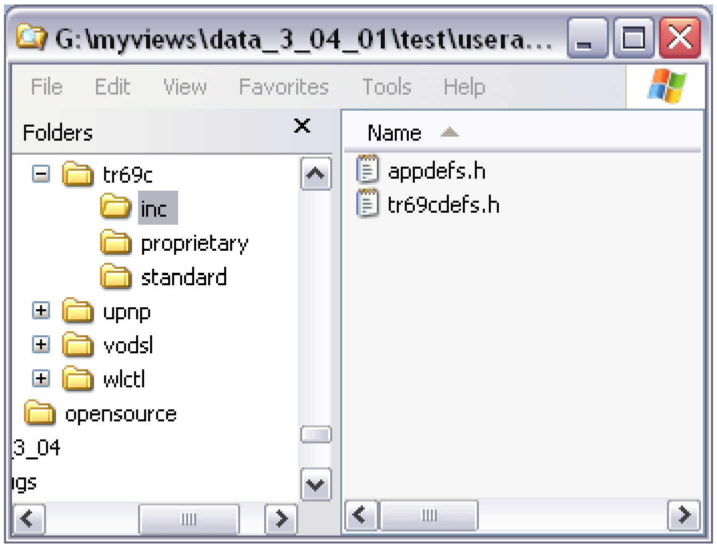


```
tr69c
    inc
    proprietary
    standard
upnp
vodsl
wlctl
```

```
inc
proprietary
standard
bcmtr69clib.o
Makefile
Makefile.bcm
tr69clib.o
```

## 3.2 tr69c/inc Directory

tr69c/inc directory should have the following files that contains header files in which defines C structures, enum, macros, and functions that are mentioned in section 2.0.

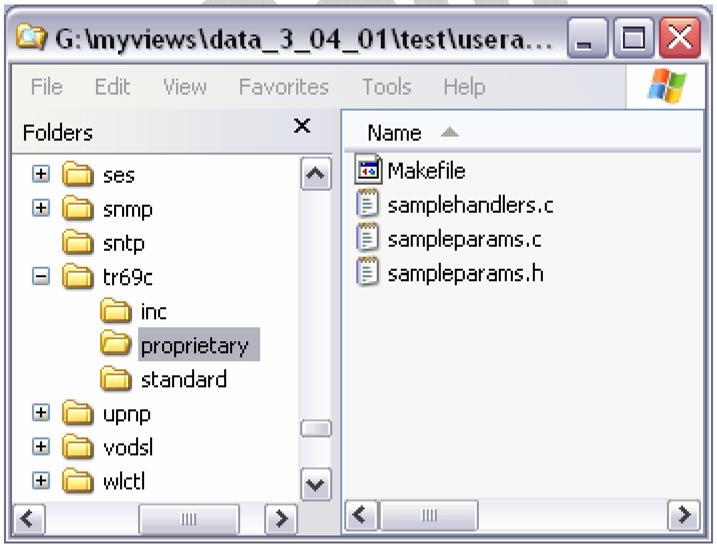

G:\myviews\data_3_04_01\test\usera...
File Edit View Favorites Tools Help
Folders
Name

tr69c
inc
proprietary
standard
upnp
vodsl
wlctl
opensource
3_04
ıgs

appdefs.h
tr69cdefs.h

## 3.3 tr69c/proprietary Directory

tr69c/proprietary directory should have the following files that provides sample codes to show how specific vendor proprietary TR-069 parameters are added to tr69c application. User should modify and provide his/her own files to this directory for adding his/her own proprietary TR-069 parameters.

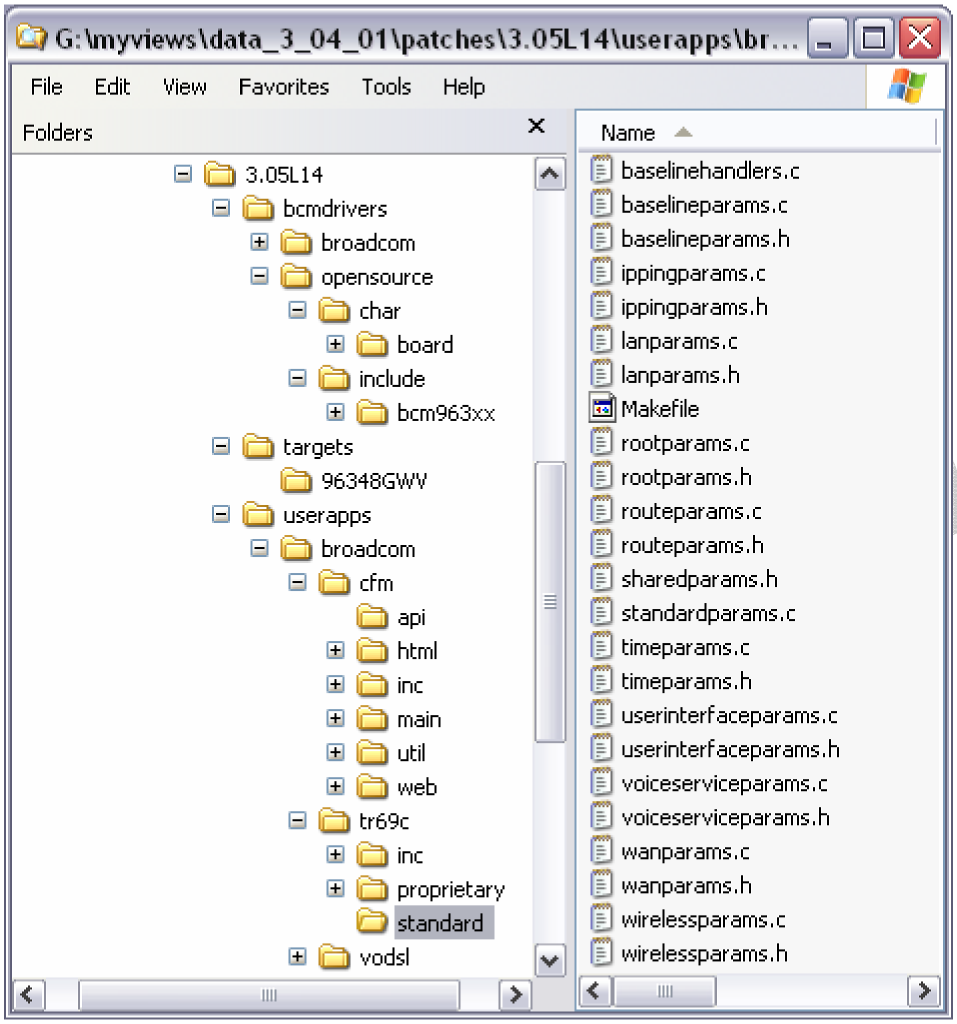

G:\myviews\data_3_04_01\test\usera...
File Edit View Favorites Tools Help
Folders
Name

ses
snmp
sntp
tr69c
inc
proprietary
standard
upnp
vodsl
wlctl

Makefile
samplehandlers.c
sampleparams.c
sampleparams.h

## 3.4 tr69c/standard Directory

tr69c/standard directory should have the following files that provides codes to allow user for customizing some existed standard TR-069 parameters, and add the new standard TR-069 parameters that are not supported.


G:\myviews\data_3_04_01\patches\3.05L14\userapps\br...
File Edit View Favorites Tools Help
Folders
Name

```
3.05L14
  bcmdrivers
    broadcom
    opensource
      char
        board
      include
        bcm963xx
  targets
    96348GWV
  userapps
    broadcom
      cfm
        api
        html
        inc
        main
        util
        web
      tr69c
        inc
        proprietary
        standard
      vodsl
```

```
baselinehandlers.c
baselineparams.c
baselineparams.h
ippingparams.c
ippingparams.h
lanparams.c
lanparams.h
Makefile
rootparams.c
rootparams.h
routeparams.c
routeparams.h
sharedparams.h
standardparams.c
timeparams.c
timeparams.h
userinterfaceparams.c
userinterfaceparams.h
voiceserviceparams.c
voiceserviceparams.h
wanparams.c
wanparams.h
wirelessparams.c
wirelessparams.h
```

## 4.0 STANDARD PARAMETERS MODIFICATION

Broadcom TR69C framework allows user to customize standard parameters under InternetGatewayDevice.DeviceInfo. by changing sources under tr69c/standard directory. User should change mainly baslinehandlers.c since it contains get/set handler functions for the following standard parameters:

InternetGatewayDevice.DeviceInfo.Manufacturer

InternetGatewayDevice.DeviceInfo.ManufacturerOUI

InternetGatewayDevice.DeviceInfo.ModelName

InternetGatewayDevice.DeviceInfo.ProductClass

InternetGatewayDevice.DeviceInfo.SerialNumber

InternetGatewayDevice.DeviceInfo.SoftwareVersion

InternetGatewayDevice.DeviceInfo.HardwareVersion

InternetGatewayDevice.DeviceInfo.SpecVersion

InternetGatewayDevice.DeviceInfo.ProvisioningCode

InternetGatewayDevice.DeviceInfo.UpTime

InternetGatewayDevice.DeviceInfo.DeviceLog

## 5.0 PROPRIETARY PARAMETERS ADDITION

This section describes with code examples the mechanism to allow user for adding vendor specific proprietary TR-069 parameters to Broadcom TR69C framework.

### 5.1 Proprietary Parameter Naming Convention

TR69C framework follows section B.2.2 Vendor-Specific Parameters in TR-069 specification for parameter naming convention with more strict.

- Company name can be either OUI or domain name. If domain name is used, only underscore is allowed (no hyphen). Ex: X_BROADCOM_COM (but not X_BROADCOM-COM).
- Proprietary parameter name MUST start with X_ to avoid any confliction with the existed parameter. Ex: X_Enable (but not Enable).

### 5.2 Top Proprietary Parameter Node

The top proprietary parameter node is the parent node of all proprietary parameters. User should add his/her proprietary parameters under this node. Below is the list of requirements for implementing the top proprietary parameter node

- It MUST be created directly under the root node “InternetGatewayDevice.”.
- Its name should be the user’s company name or OUI as mention in section 5.1 Proprietary Parameter Naming Convention.
- It should be a tObject parameter.

Section 5.3 tObject Parameter Addition describes how to add the tObject parameter. The sample codes for adding the top proprietary parameter node named X_BROADCOM_COM can be found at tr69c/standard directory in rootparams.c, and rootparams.h. It is created directly under the root node “InternetGatewayDevice” since it is added in the internetGatewayDeviceDesc[] node. Below are the sample codes

```
#ifdef PROPRIETARY
extern TRxObjNode  ProprietaryDesc[];
#endif

/* InternetGatewayDevice. */
TRxObjNode  internetGatewayDeviceDesc[] = {
…..
…..
#ifdef PROPRIETARY
   {X_BROADCOM_COM,{{tObject,0,0}}, NULL,NULL, ProprietaryDesc,NULL},
#endif
   {NULL}
};
```

The X_BROADCOM_COM TRxObjNode has the following attributes:

- X_BROADCOM_COM is its name.
- tObject is its parameter type.
- 0 is its length since it is not a string.
- 0 is its inhibitActiveNotify since it's not a counter.
- NULL is its setTRxParam since there is no set handler function associated with it.
- NULL is its getTRxParam since there is no get handler function associated with it.
- ProprietaryDesc is its objDetail which is array of TRxObjNode that describes its child nodes.
- NULL is its instanceDope. It's only used internally by TR069 framework.

## 5.3 How to Add Proprietary Parameter

Below are the general steps to add proprietary parameter to tr69c application.

1. Define the parameter name character strings using the SVAR macros in header file.
2. Declare and initialize the parameters in the parameter tree in C file. This may require adding a new parameter node linked from an existing node. If there are multiple instances of the parameter object then define an instance object node to link the new parameter to.
3. Define the prototypes for the set or get callback functions using the TRXSFUNC and TRXGFUNC macros in C file.
4. Add the necessary get or set functions to the C file. If parameter has tInstance type then also add the add and delete instance callbacks.

For each parameter type, the above steps are explained in details in the following sections.

### 5.3.1 How to Add tObject Parameter

tObject parameter is the static partial path parameter node that does not have any specific data type. It's used as partial path (internal) node that can be a parent of other full path (leaf) nodes or partial path (internal) nodes. Since it's only the partial path node, the tObject parameter should not have any get/set handler functions associated with it.

Below is the instruction to add the standard TR-069 tObject parameter named DeviceInfo.

To add a parameter, its name should be defined in header file. DeviceInfo is defined in tr69c/standard/rootparams.h using SVAR() macro as following:

```
SVAR(DeviceInfo);
```

As specify in TR-069 specification, DeviceInfo parameter should be added under InternetGatewayDevice parameter. So its object should be added in the internetGatewayDeviceDesc[] array that can be seen in tr69c/standard/rootparams.c as following:

```
 extern TRxObjNode  deviceInfoDesc [];

/* InternetGatewayDevice. */
TRxObjNode  internetGatewayDeviceDesc[] = {
   ….
   {DeviceInfo,{{tObject,0,0}}, NULL,NULL, deviceInfoDesc,NULL},
   ….
   {NULL}
};
```

The DeviceInfo TRxObjNode has the following attributes:

- DeviceInfo is its name.
- tObject is its parameter type.
- 0 is its length since it is not a string.
- 0 is its inhibitActiveNotify since it's not a counter.
- NULL is its setTRxParam since there is no set handler function associated with it.
- NULL is its getTRxParam since there is no get handler function associated with it.
- deviceInfoDesc is its objDetail which is array of TRxObjNode that describes its child nodes.
- NULL is its instanceDope. It's only used internally by TR069 framework.

### 5.3.2 How to Add tBool Parameter

tBool parameter is the full path (leaf) parameter node that has boolean as its specific data type. It's used as leaf node that can be a child of partial path (internal) nodes. Since it's the full path node that has boolean value, the tBool parameter should have either get, or set, or both handler functions associated with it.

Below is the instruction to add the proprietary TR-069 tBool parameter named X_BooleanObject under X_BROADCOM_COM parameter to create a TR-069 full path parameter named InternetGatewayDevice.X_BROADCOM_COM. X_BooleanObject.

To add a parameter, its name should be defined in header file. X_BooleanObject is defined in tr69c/proprietary/sampleparams.h using SVAR() macro as following:

```
SVAR(X_BooleanObject);
```

Since X_BooleanObject parameter should be added under X_BROADCOM_COM parameter, its object should be added in the objDetail of X_BROADCOM_COM parameter which is ProprietaryDesc [] array. This TRxObjNode array is listed in tr69c/proprietary/sampleparams.c as following:

```
TRxObjNode  ProprietaryDesc[] = {
  X_BooleanObject,{{tBool,0,0}},setBooleanObject,getBooleanObject,NULL,NULL},
    ….
   {NULL}
};
```

The X_BooleanObject TRxObjNode has the following attributes:

- X_BooleanObject is its name.
- tBool is its parameter type.
- 0 is its length since it's not string parameter.
- 0 is its inhibitActiveNotify since it's not a counter.
- setBooleanObject is its setTRxParam. It is set handler function associated with X_BooleanObject.
- getBooleanObject is its getTRxParam. It is get handler function associated with X_BooleanObject.
- NULL is its objDetail since it's the full path (leaf) node.
- NULL is its instanceDope. It's only used internally by TR069 framework.

The set handler function setBooleanObject is declared in tr69c/proprietary/sampleparams.c using TRXSFUNC macro as following:

```
TRXSFUNC(setBooleanObject);
```

The set handler function setBooleanObject is defined in tr69c/proprietary/samplehandlers.c as following

```
TRX_STATUS setBooleanObject(char *value)
{
    int status = 0;
    if ( strcmp(value, "1") == 0 )
        status = 1;
    else
        status = 0;

    return TRX_OK;
}
```

The parameter value is passed as a null terminated character string. The string value is released upon return to the framework from the setter function.

The get handler function getBooleanObject is declared in tr69c/proprietary/sampleparams.c using TRXGFUNC macro as following:

```
TRXGFUNC(getBooleanObject);
```

The get handler function getBooleanObject is defined in tr69c/proprietary/samplehandlers.c as following

```
TRX_STATUS getBooleanObject (char **value)
{
    *value = strdup("1");
    return TRX_OK;
}
```

The framework calls the function with a pointer to the value pointer. The getBooleanObject function copy the parameter value as a null terminated character string to an allocated buffer. The pointer to the buffer is written to the **value and the function return value is a TRX_STATUS value of TRX_OK, TRX_REBOOT, or TRX_ERR. The framework will release the allocated memory when it has finished building the RPC response message.

### 5.3.3 How to Add tUnsigned Parameter

tUnsigned parameter is the full path (leaf) parameter node that has unsigned integer as its specific data type. It’s used as leaf node that can be a child of partial path (internal) nodes. Since it’s the full path node that has unsigned integer value, the tUnsigned parameter should have either get, or set, or both handler functions associated with it.

Below is the instruction to add the proprietary TR-069 tUnsigned parameter named X_NumberObject under X_BROADCOM_COM parameter to create a TR-069 full path parameter named InternetGatewayDevice.X_BROADCOM_COM.X_NumberObject.

To add a parameter, its name should be defined in header file. X_NumberObject is defined in tr69c/proprietary/sampleparams.h using SVAR() macro as following:

```
SVAR(X_NumberObject);
```

Since X_NumberObject parameter should be added under X_BROADCOM_COM parameter, its object should be added in the objDetail of X_BROADCOM_COM parameter which is ProprietaryDesc [] array. This TRxObjNode array is listed in tr69c/proprietary/sampleparams.c as following:

```
TRxObjNode  ProprietaryDesc[] = {
   ….
  X_NumberObject,{{tUnsigned,0,0}},NULL,getNumberObject,NULL,NULL},
   ….
   {NULL}
};
```

The X_NumberObject TRxObjNode has the following attributes:

- `X_NumberObject` is its name.
- tUnsigned is its parameter type.
- 0 is its length since it's not string parameter.
- 0 is its inhibitActiveNotify since it's not a counter.
- NULL is its setTRxParam since there is no set handler function associated with it.
- `getNumberObject` is its getTRxParam. It is get handler function associated with X_NumberObject.
- NULL is its objDetail since it's the full path (leaf) node.
- NULL is its instanceDope. It's only used internally by TR069 framework.

The get handler function getNumberObject is declared in tr69c/proprietary/sampleparams.c using TRXGFUNC macro as following:

```
TRXGFUNC(getNumberObject);
```

The get handler function getNumberObject is defined in tr69c/proprietary/samplehandlers.c as following

```
TRX_STATUS getNumberObject(char **value)
{
        *value = strdup("1");
        return TRX_OK;
}
```

The framework calls the function with a pointer to the value pointer. The getNumberObject function copy the parameter value as a null terminated character string to an allocated buffer. The pointer to the buffer is written to the **value and the function return value is a TRX_STATUS value of TRX_OK, TRX_REBOOT, or TRX_ERR. The framework will release the allocated memory when it has finished building the RPC response message.

### 5.3.4 How to Add tString Parameter

tString parameter is the full path (leaf) parameter node that has string as its specific data type. It's used as leaf node that can be a child of partial path (internal) nodes. Since it's the full path node that has string value, the tString parameter should have either get, or set, or both handler functions associated with it.

Below is the instruction to add the proprietary TR-069 tString parameter named X_StringObject under X_BROADCOM_COM parameter to create a TR-069 full path parameter named InternetGatewayDevice.X_BROADCOM_COM.X_StringObject.

To add a parameter, its name should be defined in header file. X_StringObject is defined in tr69c/proprietary/sampleparams.h using SVAR() macro as following:

```
SVAR(X_StringObject);
```

Since X_StringObject parameter should be added under X_BROADCOM_COM parameter, its object should be added in the objDetail of X_BROADCOM_COM parameter which is ProprietaryDesc [] array. This TRxObjNode array is listed in tr69c/proprietary/sampleparams.c as following:

```
TRxObjNode  ProprietaryDesc[] = {
   ….
   {X_StringObject,{{tString,256,0}},NULL,getStringObject,NULL,NULL},
   ….
   {NULL}
};
```

The X_StringObject TRxObjNode has the following attributes:

- X_StringObject is its name.
- tString is its parameter type.
- 256 is its string length.
- 0 is its inhibitActiveNotify since it's not a counter.
- NULL is its setTRxParam since there is no set handler function associated with it.
- getStringObject is its getTRxParam. It is get handler function associated with X_StringObject.
- NULL is its objDetail since it's the full path (leaf) node.
- NULL is its instanceDope. It's only used internally by TR069 framework.

The get handler function getStringObject is declared in tr69c/proprietary/sampleparams.c using TRXGFUNC macro as following:

```
TRXGFUNC(getStringObject);
```

The get handler function getStringObject is defined in tr69c/proprietary/samplehandlers.c as following

```
TRX_STATUS getStringObject(char **value)
{
        *value = strdup("String");
        return TRX_OK;
}
```

The framework calls the function with a pointer to the value pointer. The getStringObject function copy the parameter value as a null terminated character string to an allocated buffer. The pointer to the buffer is written to the **value and the function return value is a TRX_STATUS value of TRX_OK, TRX_REBOOT, or TRX_ERR. The framework will release the allocated memory when it has finished building the RPC response message.

#### 5.3.5 How to Add tStringSOnly Parameter

tStringSOnly parameter is the full path (leaf) parameter node that has string as its specific data type. It's used as leaf node that can be a child of partial path (internal) nodes. Since it's the full path node that has string value, the tString parameter should have either get, or set, or both handler functions associated with it. Even get handler function can be associated with it, when GetParameterValues RPC method is applied to this parameter, the TR-069 framework will only return empty string, but not its actual value. This kind of parameter should be only used for object that should not be seen such as Password.

Below is the instruction to add the proprietary TR-069 tStringSOnly parameter named X_PasswordObject under X_BROADCOM_COM parameter to create a TR-069 full path parameter named InternetGatewayDevice.X_BROADCOM_COM.X_PasswordObject.

To add a parameter, its name should be defined in header file. X_PasswordObject is defined in tr69c/proprietary/sampleparams.h using SVAR() macro as following:

```
SVAR(X_PasswordObject);
```

Since X_PasswordObject parameter should be added under X_BROADCOM_COM parameter, its object should be added in the objDetail of X_BROADCOM_COM parameter which is ProprietaryDesc [] array. This TRxObjNode array is listed in tr69c/proprietary/sampleparams.c as following:

```
TRxObjNode  ProprietaryDesc[] = {
   ….
   {X_PasswordObject,{{tStringSOnly,256,0}},setPasswordObject,getPasswordObject, NULL,NULL},
   ….
   {NULL}
};
```

The X_PasswordObject TRxObjNode has the following attributes:

- X_PasswordObject is its name.
- tStringSOnly is its parameter type.
- 256 is its string length.
- 0 is its inhibitActiveNotify since it's not a counter.
- setPasswordObject is its setTRxParam. It is set handler function associated with X_PasswordObject.
- getPasswordObject is its getTRxParam. It is get handler function associated with X_PasswordObject.
- NULL is its objDetail since it's the full path (leaf) node.
- NULL is its instanceDope. It's only used internally by TR069 framework.

The set handler function setPasswordObject is declared in tr69c/proprietary/sampleparams.c using TRXSFUNC macro as following:

```
TRXSFUNC(setPasswordObject);
```

The set handler function setPasswordObject is defined in tr69c/proprietary/samplehandlers.c as following

```
TRX_STATUS setPasswordObject(char *value)
{
    // add codes to set password here
        return TRX_OK;
}
```

The parameter value is passed as a null terminated character string. The string value is released upon return to the framework from the setter function.

The get handler function getPasswordObject is declared in tr69c/proprietary/sampleparams.c using TRXGFUNC macro as following:

```
TRXGFUNC(getPasswordObject);
```

The get handler function getPasswordObject is defined in tr69c/proprietary/samplehandlers.c as following

```
TRX_STATUS getPasswordObject(char **value)
{
        *value = strdup("password");
        return TRX_OK;
}
```

The framework will return empty string and ignore the actual value that is returned from this function.

### 5.3.6 How to Add tDateTime Parameter

tDateTime parameter is the full path (leaf) parameter node that has unsigned integer as its specific data type. It's used as leaf node that can be a child of partial path (internal) nodes. Since it's the full path node that has unsigned integer value, the tDateTime parameter should have either get, or set, or both handler functions associated with it.

Below is the instruction to add the proprietary TR-069 tUnsigned parameter named X_DateTimeObject under X_BROADCOM_COM parameter to create a TR-069 full path parameter named InternetGatewayDevice.X_BROADCOM_COM.X_DateTimeObject.

To add a parameter, its name should be defined in header file. X_DateTimeObject is defined in tr69c/proprietary/sampleparams.h using SVAR() macro as following:

```
SVAR(X_DateTimeObject);
```

Since X_DateTimeObject parameter should be added under X_BROADCOM_COM parameter, its object should be added in the objDetail of X_BROADCOM_COM parameter which is ProprietaryDesc [] array. This TRxObjNode array is listed in tr69c/proprietary/sampleparams.c as following:

```
TRxObjNode  ProprietaryDesc[] = {
   ….
  X_DateTimeObject,{{tDateTime,0,0}},NULL,getDateTimeObject,NULL,NULL},
   ….
   {NULL}
};
```

The `X_DateTimeObject` TRxObjNode has the following attributes:

- `X_DateTimeObject` is its name.
- tDateTime is its parameter type.
- 0 is its length since it's not string parameter.
- 0 is its inhibitActiveNotify since it's not a counter.
- NULL is its setTRxParam since there is no set handler function associated with it.
- `getDateTimeObject` is its getTRxParam. It is get handler function associated with `X_DateTimeObject`.
- NULL is its objDetail since it's the full path (leaf) node.
- NULL is its instanceDope. It's only used internally by TR069 framework.

The get handler function getDateTimeObject is declared in tr69c/proprietary/sampleparams.c using TRXGFUNC macro as following:

```
TRXGFUNC(getDateTimeObject);
```

The get handler function getDateTimeObject is defined in tr69c/proprietary/samplehandlers.c as following

```
TRX_STATUS getDateTimeObject(char **value)
{
        static char    buf[30];
        int     c;
        struct tm   *tmp;
        time_t  curtime = time(NULL);

        tmp = localtime(&curtime);
        c = strftime(buf,sizeof(buf),"%Y-%m-%dT%H:%M:%S%z",tmp );
        memmove(&buf[c-1],&buf[c-2],3);
        buf[c-2]=':';
        *value = strdup(buf);

        return TRX_OK;
}
```

The framework calls the function with a pointer to the value pointer. The getDateTimeObject function copy the parameter value as a null terminated character string to an allocated buffer. The pointer to the buffer is written to the **value and the function return value is a TRX_STATUS value of TRX_OK, TRX_REBOOT, or TRX_ERR. The framework will release the allocated memory when it has finished building the RPC response message.

### 5.3.7 How to Add tInstance Parameter

tInstance parameter is the partial path (internal) parameter node that has a list of the same partial path parameter nodes under it. tInstance parameter has the same concept as variable that is defined as a array of structures in C language. Since it contains the list of partial path parameter nodes, the tInstance parameter should have both add, and delete handler functions associated with it to manipulate its list. It also should have another tUnsigned parameter that represents the number of partial path parameter nodes (instance nodes) under it.

Below is the instruction to add the proprietary TR-069 tInstance parameter named X_InstanceObject under X_BROADCOM_COM parameter to create a TR-069 partial path parameter named InternetGatewayDevice.X_BROADCOM_COM.X_InstanceObject.

To add tInstance parameter, another tUnsigned parameter should be added first to represent the number of partial path parameter nodes (instance nodes) under this tInstance parameter. So X_InstanceNumberOfEntries parameter is added by declaring it in tr69c/proprietary/sampleparams.h using SVAR() macro as following:

SVAR(X_InstanceNumberOfEntries);

Since X_InstanceNumberOfEntries parameter should be added under X_BROADCOM_COM parameter, its object should be added in the objDetail of X_BROADCOM_COM parameter which is ProprietaryDesc [] array. This TRxObjNode array is listed in tr69c/proprietary/sampleparams.c as following:

```
TRxObjNode  ProprietaryDesc[] = {
    ….
  X_InstanceNumberOfEntries,{{tUnsigned,0,0}},NULL, getInstanceNumberOfEntries,NULL,NULL},
    ….
   {NULL}
};
```

The set handler function is NULL since X_InstanceNumberOfEntries parameter should not be settable.

The get handler function getInstanceNumberOfEntries is declared in tr69c/proprietary/sampleparams.c using TRXGFUNC macro as following:

```
TRXGFUNC(getInstanceNumberOfEntries);
```

The get handler function getInstanceNumberOfEntries is defined in tr69c/proprietary/samplehandlers.c as following

```
extern TRxObjNode  InstanceObjectDesc[];
TRX_STATUS getInstanceNumberOfEntries(char **value)
{
        char buf[IFC_TINY_LEN];

        sprintf(buf, "%d", getInstanceCountNoPathCheck(InstanceObjectDesc));
        *value = strdup(buf);
        return TRX_OK;
}
```

Where InstanceObjectDesc is the objDetail of X_InstanceObject parameter.

To add X_InstanceObject parameter, its name should be declared in header file. X_InstanceObject is declared in tr69c/proprietary/sampleparams.h using SVAR() macro as following:

```
SVAR(X_InstanceObject);
```

Since X_InstanceObject parameter should be added under X_BROADCOM_COM parameter, its object should be added in the objDetail of X_BROADCOM_COM parameter which is ProprietaryDesc [] array. This TRxObjNode array is listed in tr69c/proprietary/sampleparams.c as following:

```
TRxObjNode  ProprietaryDesc[] = {
   ….
  X_InstanceObject,{{tInstance,0,0}},NULL,NULL,InstanceObjectDesc,NULL},
   ….
   {NULL}
};
```

The X_InstanceObject TRxObjNode has the following attributes:

- `X_InstanceObject` is its name.
- tInstance is its parameter type.
- 0 is its length since it's not string parameter.
- 0 is its inhibitActiveNotify since it's not a counter.
- NULL is its setTRxParam since there is no set handler function associated with it.
- NULL is its getTRxParam since there is no get handler function associated with it.
- InstanceObjectDesc is its objDetail. It describes more information for `X_InstanceObject.`
- NULL is its instanceDope. It's only used internally by TR069 framework.

tInstance parameter is different with other parameters in term of its objDetail field. In tInstance parameter, this field specifies more information for itself but not for other parameters under it. Since TR-069 tInstance parameter contains a list of the SAME partial path parameter nodes under it, there should be only single entry in its objDetail field. In the sample codes, the objDetail field of X_InstanceObject parameter is InstanceObjectDesc. It is listed in tr69c/proprietary/sampleparams.c as following:

```
TRxObjNode  InstanceObjectDesc[] = {
   {instanceIDMASK,{{0,0,0}},objectDelete,objectAdd,objectDesc,NULL}
};
```

The InstanceObjectDesc TRxObjNode has the following attributes:

- instanceIDMASK is its special name so that the TR-069 framework can detect this node as an object instance node by using this special name.
- 0 is its parameter type.
- 0 is its length since it's not string parameter.
- 0 is its inhibitActiveNotify since it's not a counter.
- objectDelete is its setTRxParam. It is delete handler function associated with `X_InstanceObject`.
- objectAdd is its getTRxParam. It is add handler function associated with `X_InstanceObject`.
- objectDesc is its objDetail. It describes other parameter nodes under `X_InstanceObject`.

- NULL is its instanceDope. It's only used internally by TR069 framework.

The delete handler function objectDelete is declared in tr69c/proprietary/sampleparams.c using TRXSFUNC macro as following:

```
TRXSFUNC(objectDelete);
```

The delete handler function objectDelete is defined in tr69c/proprietary/samplehandlers.c as following

```
TRX_STATUS objectDelete(char *value)
{
        TRxObjNode *n;
        InstanceDesc *idp;
        int     id = atoi(value);

        if ( (idp = findInstanceDesc(n=getCurrentNode(), id)) ) {
                if ( !deleteInstanceDesc(n, id) )
                        return TRX_OK; /* or TRX_BOOT if necessary */
        }

        return TRX_ERR;
}
```

The parameter value is passed as a null terminated character string. It is the instance ID of the instance parameter that needs to be deleted. In the delete handler function, the TRX_STATUS return is TRX_OK for object deleted, TRX_REBOOT for object deleted but requires reboot. The delete handler function should return a TRX_ERR if the object instance was not found.

The objectDelete handler function is called by the TR-069 framework when DeleteObject RPC method is applied to InternetGatewayDevice.X_BROADCOM_COM.X_InstanceObject.[i] where [i] is the instance ID of the instance parameter that needs to be deleted.

The add handler function objectAdd is declared in tr69c/proprietary/sampleparams.c using TRXGFUNC macro as following:

```
TRXGFUNC(objectAdd);
```

The add handler function objectAdd is defined in tr69c/proprietary/samplehandlers.c as following

```
TRX_STATUS objectAdd(char **value)
{
        InstanceDesc *idp;

        if ( (idp = getNewInstanceDesc(getCurrentNode(), getCurrentInstanceDesc(), 0)) ) {
            char buf[IFC_TINY_LEN];
            sprintf(buf, "%d", idp->instanceID);
            *value = strdup(buf);
            return TRX_OK; /* or TRX_BOOT if necessary */
        }

        return TRX_ERR;
}
```

In the add handler function, the new instance value, as a null terminated string, is returned in the **value variable. The TRX_STATUS return is TRX_OK for object added successfully, TRX_REBOOT for object deleted but requires reboot. If there is an error in adding the object the add handler function should return TRX_ERR with any allocated memory freed.

The objectDelete handler function is called by the TR-069 framework when DeleteObject RPC method is applied to InternetGatewayDevice.X_BROADCOM_COM.X_InstanceObject.[i] where [i] is the instance ID of the instance parameter that needs to be deleted.

### 5.3.8 How to Initialize tInstance Parameter

The list of the SAME instance parameter nodes under tInstance parameter is implemented as a link list in TR-069 framework. Each instance parameter node in this list is described by an InstanceDesc structure that is linked to the tInstance parameter node. An instance descriptor parameter node is described by:

```
typedef struct InstanceDesc {
   struct      InstanceDesc *next;
   int         instanceID;
   void        *hwUserData;
} InstanceDesc;
```

The instanceID is created by the AddObject RPC method. The specification requires that the instanceID numbers not be reused until the integer number range wraps. When tr69c starts, the instance descriptor pointer in each instance parameter node is initialized with the list of currently valid InstanceDesc. The hwUserData is a void pointer that can be used by the specific instance routines as a handle to configuration specific data.

The new instance is created using the getNewInstanceDesc() function that is declared in tr69c/inc/tr69cdefs.h as following:

```
extern InstanceDesc *getNewInstanceDesc( TRxObjNode *n, InstanceDesc *parent, int id);
```

where

- n is the instance parameter node under tInstance parameter node.
- parent is the instance descriptor of the parent of the tInstance parameter node .
- id is the instance ID number of the current instance parameter node.

TR-069 framework provides a function in which the codes to initialize proprietary tInstance parameters should be added. This initProprietaryInstance() function is called when tr69c is started. It is listed in tr69c/proprietary/samplehandlers.c as following

```
void initX_InstanceObject(InstanceDesc *proprietaryInstanceDesc) {
        TRxObjNode *n = InstanceObjectDesc;
        InstanceDesc *idp = NULL;
        int i = 0;

        /* Initialize InternetGatewayDevice.X_BROADCOM_COM.X_InstanceObject.{i}. */
        for ( i = 1; i <= 2; i++ ) {
            if ( findInstanceDescNoPathCheck(n, i) == NULL ) {
                        idp = getNewInstanceDesc(n, proprietaryInstanceDesc, i);
                        idp->instanceID = i;
                        idp->hwUserData = NULL;
              }
        }
}
extern TRxObjNode  ProprietaryDesc[];
 void initProprietaryInstance(void) {
        /* There is only a single Proprietary Instance on this design */
        TRxObjNode *n = ProprietaryDesc;
        int id = 1;
        InstanceDesc *proprietaryInstanceDesc = getNewInstanceDesc(n, NULL, id);

        initX_InstanceObject(proprietaryInstanceDesc);
}
```