OpenClovis
Software Development Kit (SDK)
Service Description and API Reference for Mediation Management Service

For OpenClovis SDK Release 2.3 V0.4

Document Revision Date: March 08, 2007

Contents

Chapter 1

Functional Overview

Customer systems can be managed by different external managers like SNMP, CLI, CMIP, XML, WEB, and so on. These managers communicate to their corresponding agents in the system using the specific protocol. The management agents in a system have to translate the attributes and/or commands into OpenClovis run-time environment attributes and operations.

The OpenClovis Mediation component acts like a gateway into the OpenClovis run-time environment on the System Controller. On one side, it interacts with management agent like, SNMP agent, CLI agent, and on the other side it interacts with OpenClovis components or managers. The primary responsibility of the Mediation module is to translate the managed service requests from the management agent into requests in the OpenClovis run-time environment.

Mediation component runs as a library in the context of a management agent. It maintains the translation between a specific external management identifier and OpenClovis run-time environment identifier(s). It also maintains the translation between an external operation and corresponding COR operation.

This component also allows the management agents to register for asynchronous notifications within the system. All the notifications are modeled as alarm in COR.

When SNMP agent is being destroyed (probably as part of system shut down,) it can destroy the mediation component by calling `clMedFinalize(gMedHdl)`.

Chapter 2

Service Model

TBD

Chapter 3

Service APIs

3.1 Type Definitions

3.1.1 ClMedHdlPtrT

typedef void ClMedHdlPtrT;*

An identifier for the mediation module.

3.1.2 ClMedAgentId

```
typedef struct {
        ClUint8T *id;
        ClUint32T len;
} ClMedAgentId;
```

The structure `ClMedAgentId` contains the external management specific representation of the identifier(s). The attributes of the structure are:

- *id* - Identifier in the agent language. Example, for SNMP, OID are considered as the ID.
- *len* - Length of the identifier.

3.1.3 ClMedNotifyHandlerCallbackT

```
typedef ClRcT(*ClMedNotifyHandlerCallbackT)(
        struct ClMedAgentId *,
        ClUint32T,
        ClCorMOIdPtrT,
        ClUint32T,
        char *,
        ClUint32T
);
```

The type of a callback function that is used to notify a trap/change. The attributes of the function are:

- *ClMedAgentId* - External Agent ID.
- *ClUint32T* - Session ID. Currently not in use.
- *ClCorMOIdPtrT* - MOID of the COR object where the alarm is raised.
- *ClUint32T* - Probable cause of the alarm.
- *char* - Optional data.
- *ClUint32T* - Data length.

3.1.4 ClMedInstXlatorCallbackT

```
typedef ClRcT(*ClMedInstXlatorCallbackT)(
        const struct ClMedAgentId *,
        ClCorMOIdPtrT hmoId,
        ClCorAttrPathPtrT containedPath,
        void **pInstance,
        ClUint32T *instLen,
        ClUint32T create
);
```

The type of the callback function that acts as an instance translator for COR managed attributes. This is called when a new object is created. If `create` has a value 0, it returns back the index for the table represented by the COR MO. If it is non-zero, index is passed to it and it returns the MOID of the corresponding COR object. The parameters of this function are:

- *ClMedAgentId* - External Agent ID.
- *hmoId* - If `create` is equal to 0, MOID of the COR object is passed for which the index needs to be generated. If create is non-zero, index is passed as the input and MOID is returned.
- *containedPath* - Pointer to containment path.
- *pInstance* - Instance in external agent format.
- *instLen* - Instance length.
- *create* - Specifies if the index for a given MOID (create = 0) should be generated or if the MOID for a given index (create is non-zero) should be generated.

3.1.5 ClMedVarBindT

```
typedef struct {
        ClMedAgnetIdT attrId;
        void *pVal;
        ClUint32T len;
        void *outBuf;
        ClUint32T outBufLen;
        void *pInst;
        ClUint32T instLen;
        ClUint32T errId;
```

```
        void *cookie;
        ClUint32T index;
        ClCorAttrIdT ContAttrId;
} ClMedVarBindT;
```

The structure, `ClMedVarBindT`, contains the information required to perform an external operation. The attributes of the structure are:

- *attrId* - Attribute ID in external agent format.
- *pVal* - Value that is received for GET operation or which is passed for SET operation.
- *len* - Length of the value.
- *outBuf* - Output buffer, currently not used.
- *OutBufLen* - Length of the output buffer. Currently not used.
- *pInst* - Index value.
- *instLen* - Length of the instance.
- *errId* - Error ID returned for the operation.
- *cookie* - Any additional information that needs to be passed for the operation.
- *index* - Array index, if the attribute is of type array. Otherwise, it should be set to -1.
- *ContAttrId* - Containment attribute ID.

3.1.6 ClMedTgtOpCodeT

```
typedef struct {
        ClMedTgtTypeEnumT type;
        ClMedCorOpEnumT opCode;
} ClMedTgtOpCodeT;
```

The structure, `ClMedTgtOpCodeT`, contains the target operation code. The attributes of the structure are:

- *type* - Type of operation. This can be a COR or non COR operation. Currently, COR operations are supported.
- *opCode* - `OpCode` in COR format.

3.1.7 ClMedOpT

```
typedef struct {
        ClUint32T opCode;
        ClUint32T varCount;
        ClMedVarBindT *varInfo;
} ClMedOpT;
```

The structure, `ClMedOpT`, contains the operation information. The attributes of the structure are:

- *opCode* - Operation Code of the external agent.
- *varCount* - Number of variables for this operation.
- *varInfo* - Pointer to an array of operands.

3.1.8 ClMedCorIdT

```
typedef struct {
        ClCorMoIdPtrT moId;
        ClCorAttrIdT* attrId;
} ClMedCorIdT;
```

The structure, `ClMedCorIdT`, represents COR identifier. The attributes of the structure are:

- *moId* - MOID of the object containing the attribute.
- *attrId* - Attribute identifier.

3.1.9 ClMedCorOpEnumT

```
typedef enum {
        CL_COR_SET=1,
        CL_COR_GET,
        CL_COR_GET_NEXT,
        CL_COR_CREATE,
        CL_COR_DELETE,
        CL_COR_WALK,
        CL_COR_GET_FIRST,
        CL_COR_GET_SVCS,
        CL_COR_CONTAINMENT_SET,
        CL_COR_MAX,
} ClMedCorOpEnumT;
```

The enumeration, `ClMedCorOpEnumT`, contains the various COR operations. The attributes of the enumeration are:

- *CL_COR_SET* - Set operation.
- *CL_COR_GET* - Get operations.
- *CL_COR_GET_NEXT* - Get next operations.
- *CL_COR_CREATE* - Create operation.
- *CL_COR_DELETE* - Delete operation.
- *CL_COR_WALK* - Get all operations.
- *CL_COR_GET_FIRST* - Get first operation.
- *CL_COR_GET_SVCS* - Get all the services for a particular COR MO.
- *CL_COR_CONTAINMENT_SET* - Set operation for a containment attribute.
- *CL_COR_MAX* - Invalid.

3.1.10 ClMedTgtTypeEnumT

```
typedef enum {
        CL_MED_XLN_TYPE_COR=1,
        CL_MED_XLN_TYPE_NON_COR
} ClMedTgtTypeEnumT;
```

The enumeration, `ClMedTgtTypeEnumT` is the type of an identifier for the target. The attributes of the enumeration are:

- *CL_MED_XLN_TYPE_COR* - Identifier is a COR attribute.
- *CL_MED_XLN_TYPE_NON_COR* - Identifier is not a COR attribute.

3.1.11 ClMedTgtIdT

```
typedef struct {
        ClMedTgtTypeEnumT type;
        ClUint16T eoId;
        union
        {
            ClMedCorIdT corId;
            ClMedSysIdT sysId;
        }info
} ClMedTgtIdT;
```

The structure, `ClMedTgtIdT`, contains the type of a target identifier. The attributes of the structure are:

- *type* - Target identifier type.
- *eoId* - EO identifier.

The attributes of the union, `info` are:

- *corId* - Identifier in COR terms.
- *sysId* - Identifier in non-COR terms. Currently, this is not supported.

3.1.12 ClMedAgntOpCodeT

```
typedef struct {
        ClUint32T opCode;
} ClMedAgntOpCodeT;
```

The structure, `ClMedAgntOpCodeT`, contains the operation code of the management agent. The attributes of the structure are:

- *opCode* - `OpCode` in native form.

3.2 Library Life Cycle APIs

3.2.1 clMedInitialize

clMedInitialize

Synopsis:
Initializes the Mediation Library.

Header File:
clMedApi.h

Syntax:

```
ClRcT  clMedInitialize(
                          CL_OUT ClMedHdlPtrT       *medHdl,
                          CL_IN ClMedNotifyHandlerCallbackT  fpTrapHdlr,
                          CL_IN ClMedInstXlatorCallbackT  instXlator,
                          CL_IN ClCntKeyCompareCallbackT fpInstCompare);
```

Parameters:
medHdl: (out) Handle to mediation.

fpTrapHdlr: (in) Function pointer to handle the notifications.

instXlator: (in) Function pointer to handle the instance translations.

fpInstCompare: (in) Function pointer to compare two instances of objects.

Return values:
CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `medHdl` contains a NULL pointer.

CL_ERR_NO_MEMORY: Memory allocation failure.

Description:
This function is used to initialize the Mediation Library. The callbacks for notification handling, instance translation, and key comparison e to be provided. As part of the initialization, this function returns the mediation handle. This function is invoked before any operation on the Mediation Library is performed.

Library File:
libClMedClient.a, libClMedClient.so

Related Function(s):
clMedFinalize

3.2.2 clMedFinalize

clMedFinalize

Synopsis:
Destroys the Mediation Library.

Header File:
clMedApi.h

Syntax:

```
ClRcT  clMedFinalize(
                        CL_IN ClMedHdlPtrT medHdl);
```

Parameters:
medHdl: (in) Handle returned as part of initialization.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_PARAMETER: An invalid parameter has been passed to the function. A parameter is not set correctly.

Description:
This function is used to free or finalize the Mediation Library. This must be supplied with a valid mediation handle obtained during initialization of the Mediation Library. This is called when further usage of the Mediation Library is not required. The container lists are destroyed and the event channels are closed as part of this function.

Library File:
libClMedClient.a, libClMedClient.so

Note:
This function is invoked as a part of mediation clean up.

Related Function(s):
clMedInitialize

3.3 Functional APIs

3.3.1 clMedOperationExecute

clMedOperationExecute

Synopsis:
Executes a COR operation for the corresponding external operation.

Header File:
clMedApi.h

Syntax:

```
ClRcT  clMedOperationExecute(
                                  CL_IN ClMedHdlPtrT     medHdl,
                                  CL_INOUT ClMedOpT      *opInfo);
```

Parameters:
medHdl: (in) Handle returned as part of initialization.

opInfo: (in/out) Pointer to an array of type `ClMedOpT`.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_PARAMETER: An invalid parameter has been passed to the function. A parameter is not set correctly.

CL_MED_ERR_NO_OPCODE: The `opCode` is not added.

Description:
This function is used to execute an operation corresponding to an external operation. The Mediation Library is an interface to COR. It takes the external operation, converts it into the corresponding COR operation, and invokes COR APIs to perform the operation.

Library File:
libClMedClient.a

Note:
This function is invoked in order to execute a COR operation.

Related Function(s):
None.

3.3.2 clMedIdentifierTranslationInsert

clMedIdentifierTranslationInsert

Synopsis:
Adds an entry to the identifier translation table.

Header File:
clMedApi.h

Syntax:

```
ClRcT  clMedIdentifierTranslationInsert(
                            CL_IN ClMedHdlPtrT      medHdl,
                            CL_IN ClMedAgnetIdT   clientIdentifier,
                            CL_IN ClMedTgtIdT    *corIdentifier,
                            CL_IN ClUint32T    elemCount);
```

Parameters:
medHdl: (in) Handle returned as part of initialization.

clientIdentifier: (in) Identifier in external agent language.

corIdentifier: (in) Array of COR Identifiers.

elemCount: (in) Number of COR identifiers.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_PARAMETER: An invalid parameter has been passed to the function. A parameter is not set correctly.

CL_ERR_NO_MEMORY: Memory allocation failure.

Description:
This function is used to add an entry to the identifier translation table. The entries are added based on the external client identifier value.

Library File:
libCIMedClient.a, libCIMedClient.so

Related Function(s):
None.

3.3.3 clMedIdentifierTranslationDelete

clMedIdentifierTranslationDelete

Synopsis:
Deletes an entry from the identifier translation table.

Header File:
clMedApi.h

Syntax:

```
ClRcT  clMedIdentifierTranslationDelete(
                            CL_IN ClMedHdlPtrT       medHdl,
                            CL_IN ClMedAgnetIdT    clientIdentifier);
```

Parameters:
medHdl: (in) Handle returned as part of initialization.

clientIdentifier: (in) Identifier in agent language.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_PARAMETER: An invalid parameter has been passed to the function. A parameter is not set correctly.

Description:
This function is used to delete an entry from the identifier translation table. The mediation manager adds the entry based on the client identifier. The entry is deleted when a valid mediation handle with a correct client identifier is provided.

Library File:
libClMedClient.a, libClMedClient.so

Related Function(s):
clMedIdentifierTranslationInsert

3.3.4 clMedOperationCodeTranslationAdd

clMedOperationCodeTranslationAdd

Synopsis:
Adds an entry to the identifier translation table.

Header File:
clMedApi.h

Syntax:

```
ClRcT  clMedOperationCodeTranslationAdd(
                                  CL_IN ClMedHdlPtrT medHdl,
                                  CL_IN ClMedAgntOpCodeT agntOpCode,
                                  CL_IN ClMedTgtOpCodeT *tgtOpCode,
                                  CL_IN ClUint32T opCount);
```

Parameters:

medHdl: (in) Handle returned as part of initialization.

agntOpCode: (in) Operation identifier in external agent terminology.

tgtOpCode: (in) Operation identifier in COR terminology.

opCount: (in) Operation count.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_PARAMETER: An invalid parameter has been passed to the function. A parameter is not set correctly.

CL_ERR_NO_MEMORY: Memory allocation failure.

Description:
This function is used to add an entry to the Operation code translation table for the agent. This is similar to the mediation identifier translation table. The agent `opCode` must have a corresponding COR `opCode` for adding it into the `opCode` translation table.

Library File:
libClMedClient.a,libClMedClient.so

Related Function(s):
clMedIdentifierOperationCodeTranslationDelete

3.3.5 clMedOperationCodeTranslationDelete

clMedOperationCodeTranslationDelete

Synopsis:
Deletes an entry from the operation translation table.

Header File:
clMedApi.h

Syntax:

```
ClRcT  clMedOperationCodeTranslationDelete(
                                  CL_IN ClMedHdlPtrT         medHdl,
                                  CL_IN ClMedAgntOpCodeT  agntOpCode);
```

Parameters:
medHdl: (in) Handle returned as part of initialization.

agntOpCode: (in) Operation identifier in agent terminology.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_PARAMETER: An invalid parameter has been passed to the function. A parameter is not set correctly.

Description:
This function is used to delete an entry from the operation translation table. To delete an entry from the `opCode` table, a valid mediation handle and a valid agent `opCode` are required.

Library File:
libClMedClient.a,libClMedClient.so

Related Function(s):
clMedIdentifierTranslationInsert

3.3.6 clMedErrorIdentifierTranlationInsert

clMedErrorIdentifierTranlationInsert

Synopsis:
Adds an entry to the error ID translation table.

Header File:
clMedApi.h

Syntax:

```
ClRcT  clMedErrorIdentifierTranlationInsert(
                                CL_IN ClMedHdlPtrT    medHdl,
                                CL_IN ClUint32T  sysErrorType,
                                CL_IN ClUint32T  sysErrorId,
                                CL_IN ClUint32T  agntErrorId);
```

Parameters:
medHdl: (in) Handle returned as part of initialization.

sysErrorType: (in) External error type.

sysErrorId: (in) External error ID.

agntErrorId: (in) Corresponding COR error ID.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_PARAMETER: An invalid parameter has been passed to the function. A parameter is not set correctly.

CL_ERR_NO_MEMORY: Memory allocation failure.

Description:
This function is used to add an entry to the `ErrorId` translation table. The COR error ID is supplied with the corresponding external error ID and external error type. The function then adds the entry of `sysErrorId` and `sysErrorType` for the value of the `agentErrorId`.

Library File:
libClMedClient.a,libClMedClient.so

Related Function(s):
clMedErrorIdentifierTranlationDelete

3.3.7 clMedErrorIdentifierTranlationDelete

clMedErrorIdentifierTranlationDelete

Synopsis:
Deletes an entry from the error ID translation table.

Header File:
clMedApi.h

Syntax:

```
ClRcT  clMedErrorIdentifierTranlationDelete(
                                CL_IN ClMedHdlPtrT    medHdl,
                                CL_IN ClUint32T  sysErrorType,
                                CL_IN ClUint32T  sysErrorId);
```

Parameters:

medHdl: (in) Handle returned as part of initialization.

sysErrorType: (in) System error type.

sysErrorId: (in) System error ID.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_PARAMETER: An invalid parameter has been passed to the function. A parameter is not set correctly.

Description:
This function is used to delete an entry from the error ID translation table. A value of the agent `errorId` and mediation handle must be provided for the entry in the error identifier table to be deleted.

Library File:
libClMedClient.a,libClMedClient.so

Related Function(s):
clMedErrorIdentifierTranlationInsert

Chapter 4

Service Management Information Model

TBD

Chapter 5

Service Notifications

TBD

Chapter 6

Debug CLIs

TBD

Index