OpenClovis
Software Development Kit (SDK)
Service Description and API Reference for Remote Method Dispatch (RMD) Service

For OpenClovis SDK Release 2.3 V0.4

Document Revision Date: March 13, 2007

Contents

Chapter 1

Functional Overview

OpenClovis ASP is a distributed system with many redundant servers. These servers interact with each other to provide the required services to the applications. The servers can be individual processes executing in separate address spaces. The interaction is possible, if an efficient and effective means of communication exists between these servers. Remote Procedure Call (RPC) mechanism is often used to enable communication between processes existing in different address spaces.

The OpenClovis Remote Method Dispatch (RMD), implements RPC semantics using synchronous and asynchronous methods. RMD transparently invokes methods exported by various objects. The objects on which a method is invoked can exist locally or on a remote blade. RMD supports both synchronous and asynchronous calls. The latter includes calls with and without return information. Signature check and version check is performed at run-time as part of the RMD call. At-most-once semantics ensures that the request is never executed more than once by the service provider.

RMD calls can fail due to reasons beyond the semantics of the operation. Some of the possible causes include link failure, component failure, and communication overload. These errors are handled by the RMD and the error handlers registered by the application.

Chapter 2

Service Model

2.1 Usage Model

When an object requires the service of another object, it needs to call that particular object. This call is trapped by the RPC mechanism, which sends the parameters of the call to the destination object across the network. The RPC mechanism receives the parameters and calls the relevant function residing on the destination server. This function executes in the address space of the destination and returns the control (and parameters) to the RPC mechanism, on the server at the destination. This return result is returned to the RPC mechanism at the client (caller). The caller's RPC returns the result to the caller, that continues as if it returned from a normal (local) function call. The RMD is built on the same mechanism as the RPC. The RMD mechanism uses the OpenClovis IOC protocol as the transport layer for communication and enables communication between components (processes). IOC is responsible for the transportation of packets across servers. A component is the unit of execution of ASP and it is equivalent to a process in many ways.

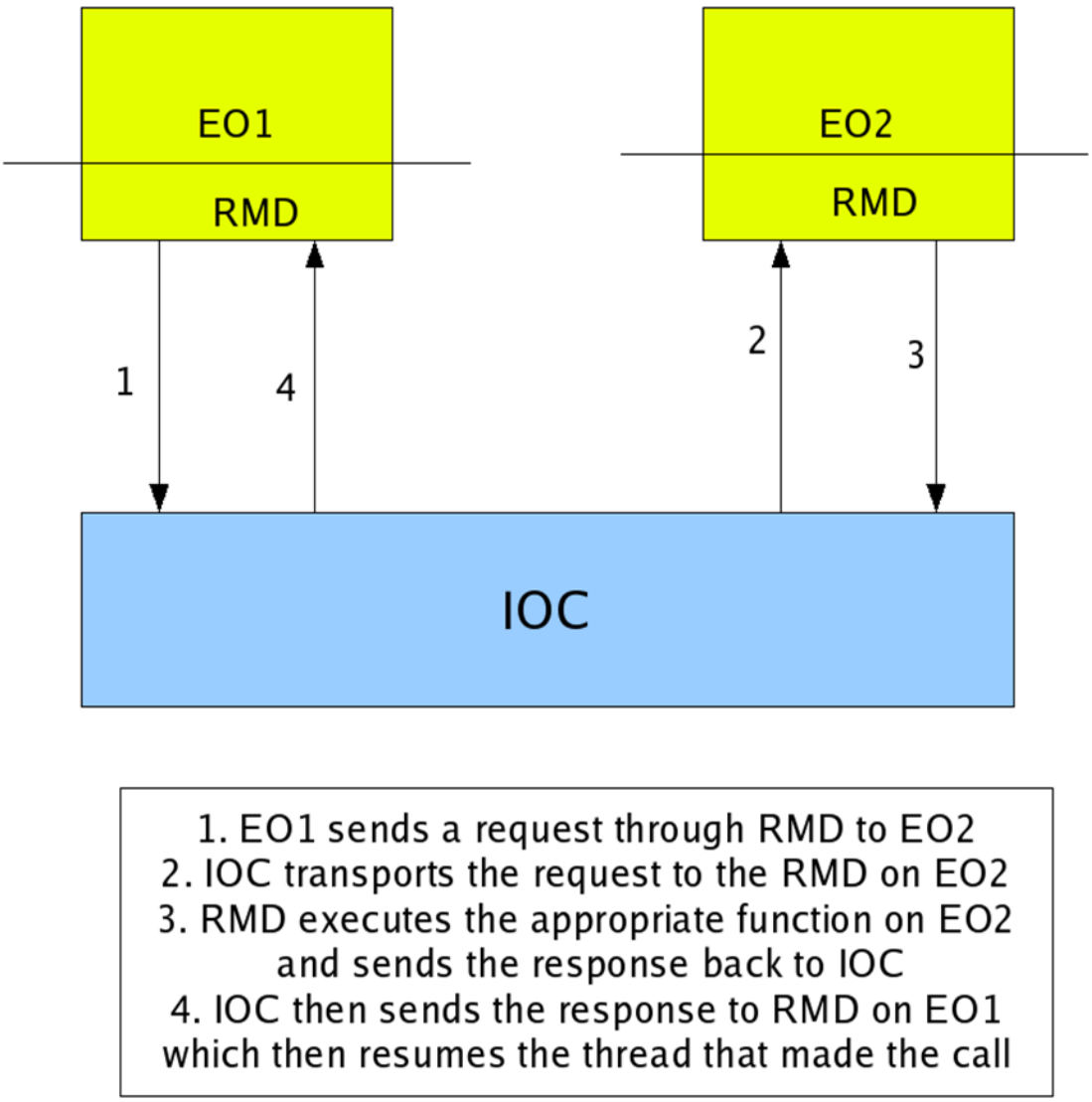

Figure 2.1: Communication Based on RMD

Chapter 3

Service APIs

3.1 Type Definitions

3.1.1 ClRmdOptionsT

typedef struct ClRmdOptions{
ClUint32T timeout;
ClUint32T retries;
ClUint8T priority;
ClIocToBindHandleT transportHandle;
} ClRmdOptionsT;

The structure, `ClRmdOptionsT`, is used to pass any optional parameters to RMD.

- *timeout* - Timeout value in milliseconds. It is the expected time of execution of the remote function in the order of the minimum resolution provided by the timer library. The values 1 or 0 indicate timeout forever.
- *retries* - Maximum number of times you can retry after the first call, if a timeout has occurred.
- *priority* - Priority value for the call. It is passed to the IOC without modification.
- *transportHandle* - Transport handle obtained through `clIocBind()` for an RMD call through the specified transport.

3.1.2 clRMDAsyncOptionsT

typedef struct {
ClRmdAsyncCallbackT fpCallback;
*void *pCookie;*
} clRMDAsyncOptions;

typedef struct clRMDAsyncOptions ClRmdAsyncOptionsT;

This structure, `clRMDAsyncOptions`, is used to pass additional parameters to the `clRmdWithMsg()` asynchronous call.

- *fpCallback* - Calling application's callback.
- *pCookie* - Calling application's cookie.

3.2 Functional APIs

3.2.1 clRmdWithMsg

clRmdWithMsg

Synopsis:
Invokes a remote function call when the parameters are passed as messages.

Header File:
clRmdApi.h

Syntax:

```
ClRcT clRmdWithMsg(
                          ClIocAddressT        remoteObjAddr,
                          ClUint32T           funcId,
                          ClBufferHandleT    inMsgHdl,
                          ClBufferHandleT    outMsgHdl,
                          ClUint32T           flags,
                          ClRmdOptionsT*       pOptions,);
```

Parameters:

remoteObjAddr: Address of the destination Object.

funcId: Function ID of the function to be executed.

inMsgHdl: This parameter is created and freed by the calling application, if `CL_RMD_CALL_NON_PERSISTENT` is not set. If this flag is set, RMD frees the message. If this parameter is NULL, no value is passed to the remote end by the caller.

outMsgHdl: Created and freed by the caller. If it is NULL and `CL_RMD_CALL_NEED_REPLY` flag is set, `CL_RMD_RC (CL_ERR_INVALID_PARAM)` is returned.

flags: Informs RMD if the call is a synchronous call or an asynchronous call. It also indicates if at-most-once semantics is required or not.

options: Indicates the optional parameters such as priority, timeout, cookie, retries, and callback function. If it is NULL, default values (priority and timeout) are taken.

pAsyncOptions: This is to be passed only if the call is an asynchronous call. Optional parameters such as cookies and callback functions can be passed in this parameter.

Return values:

CL_OK: The function executed successfully.

CL_RMD_RC(CL_ERR_NO_MEMORY): The system memory is not available.

CL_RMD_RC(CL_ERR_TIMEOUT): Reply to this call was not received in the specified time or an invalid IOC port ID is passed to the function.

CL_RMD_RC(CL_ERR_NULL_PTR): `pOptions` contains a NULL pointer.

CL_RMD_RC(CL_INVALID_PARAMETER): An invalid parameter is passed as a parameter to the function. A parameter is not set correctly.

CL_EO_ERR_FUNC_NOT_REGISTERED: The requested function is not registered.

CL_EO_ERR_EO_SUSPENDED: The remote component is in the suspended state.

CL_RMD_RC(CL_ERR_NOT_INITIALIZED): RMD Library is not initialized.

CL_IOC_RC(CL_IOC_ERR_RECV_UNBLOCKED): The receiver is unblocked.

CL_IOC_RC(CL_IOC_ERR_HOST_UNREACHABLE): `remoteObjAddr` contains an invalid node address.

CL_IOC_RC(CL_ERR_NOT_EXIST): An invalid logical address has been passed to this function.

CL_RMD_RC(CL_ERR_NOT_IMPLEMENTED): This type of call is not supported. This function can also return some OS defined error codes.

Description:

This function is used to invoke a remote function call. The following parameters are required:

- Destination address (`ObjectId`) where the function resides
- `functionId` of the function to be invoked
- Input parameter in a message
- Output message to receive the reply
- Flags specific to RMD
- RMD options
- RMD asynchronous call options - For an asynchronous call, options such as priority, timeout value, retries, cookie, and callback function must be passed in the structure, `ClRmdAsyncOptions`.

All parameters are not mandatory for every call. For a synchronous call, cookie and callback functions are not required. The remote function is identified by the remote object address and the function ID.

Library File:

libClRmd

Related Function(s):

None

3.2.2 clRmdLibInitialize

clRmdLibInitialize

Synopsis:
Initializes the RMD library.

Header File:
clRmdApi.h

Syntax:

```
ClRcT clRmdLibInitialize(void);
```

Return Values:

CL_OK: The function executed successfully.

CL_RMD_RC(CL_ERR_NO_MEMORY): System memory is not available.

CL_RMD_RC(CL_ERR_NOT_INITIALIZED): RMD Library is not initialized.

Description:
This function initializes the RMD library with default values if the RMD library is not initialized as part of the configuration of the component. Before an RMD function can be used, this function must be executed.

Library File:
libClRmd

Related Function(s):
clRmdLibFinalize.

3.2.3 clRmdLibFinalize

clRmdLibFinalize

Synopsis:
Finalizes the RMD library.

Header File:
clRmdApi.h

Syntax:

```
ClRcT clRmdLibFinalize(void);
```

Return Values:

CL_OK: The function executed successfully.

CL_RMD_RC(CL_ERR_NO_MEMORY): System memory is not available.

CL_RMD_RC(CL_ERR_NOT_INITIALIZED): RMD Library is not initialized.

Description:
This function is used to free the resources acquired by the RMD library when the RMD library is initialized. It must be invoked typically during the system shutdown process or if the RMD library is not required.

Library File:
libClRmd

Related Function(s):
clRmdLibInitialize.

Index