OpenClovis
Software Development Kit (SDK)
Service Description and API Reference for Execution Object (EO) Service

For OpenClovis SDK Release 2.3 V0.4

Document Revision Date: March 27, 2007

Contents

Chapter 1

Functional Overview

The OpenOpenClovis Execution Object (EO) encapsulates each distinct OpenOpenClovis ASP aware software component and provides an execution environment for the components. It provides a uniform interface between the software component and the rest of the system components. The interfaces fall into the following two categories:

- Management Interface - This interface is used to control and configure the software components.
- Service Interface - This interface allows software components to expose component specific functionality.

  Both management and service interfaces are exposed using RMD APIs. EO provides threads for receiving RMD messages and worker threads to process them. It provides an execution environment, required by a software component, to the component user and component manager.

  The OpenClovis product suite provides a process of integrating a third party software component with OpenClovis ASP. This process is known as Componentization. Using Componentization, both management and service interfaces are exposed through RMD.

  Componentization provides the following functionality:

  - Component re-start
  - Service Migration
  - Location Transparency
  - Easy debugging, statistics gathering, and profiling

  Componentization helps in features such as:

  - Resource Management
  - Component start, stop, and restart
  - Debugging

EO communicates to other components using the OpenClovis Communication Core components such as Event Manager (EM), Remote Method Dispatch (RMD), Intelligent Object Communication (IOC), and Name Service.

Chapter 2

Service Model

TBD

Chapter 3

Service APIs

3.1 Type Definitions

3.1.1 clEoExecutionObj

typedef struct {

textitClEoAppCreateCallbackT clEoCreateCallout;
ClEoAppDeleteCallbackT clEoDeleteCallout;
ClEoAppHealthCheckCallbackT clEoHealthCheckCallout;
ClEoAppStateChgCallbackT clEoStateChgCallout;
ClIocCommPortHandleT commObj;
ClEoIdT eoID;
ClUint32T eoInitDone;
ClOsalMutexIdT eoMutex;
ClIocPortT eoPort;
ClUint32T eoSetDoneCnt;
ClCntHandleT eoTaskIdInfo;
ClUint32T maxNoClients;
ClCharT name [CL_EO_MAX_NAME_LEN];
ClUint32T noOfThreads;
*ClEoClientObjT *pClient;*
ClCntHandleT pEOPrivDataHdl;
ClOsalThreadPriorityT pri;
ClUint32T refCnt;
ClRmdObjHandleT rmdObj;
ClEoStateT state;
ClUint32T threadRunning;
} clEoExecutionObj;

The structure, `clEoExecutionObj`, contains the properties of an EO execution object. These properties constitute the properties of running OS thread or process.

- *appType* - Indicates if the application needs the main thread.
- *clEoCreateCallout* - Application function that is called from `main()` during the initialization process.

- *clEoDeleteCallout* - Application function that is called when the EO is terminated.
- *clEoHealthCheckCallout* - Application function that is called when EO health check is performed by CPM.
- *clEoStateChgCallout* - Application function that is called when the EO is moved into the suspended state.
- *commObj* - EO communication object.
- *eoID* - Unique EOID of a blade.
- *eoInitDone* - This indicates if `EOInit()` has been called.
- *eoMutex* - Mutex that is used to protect the Execution Object.
- *eoPort* - Requested IOC Communication Port.
- *eoSetDoneCnt* - Used to set state related flag and counter.
- *eoTaskIdInfo* - TaskID information of receive loop. It is used to delete the EO.
- *maxNoClients* - Maximum number of EO clients.
- *name[CL_EO_MAX_NAME_LEN]* - Execution object name.
- *noOfThreads* - Number of RMD threads spawned.
- *pClient* - Pointer to EO client functions.
- *pEOPrivDataHdl* - Handle of the container of EO specific data.
- *pri* - Priority of the EO threads where RMD is executed.
- *rmdObj* - RMD object associated with the state of the EO.
- *threadRunning* - State of the receive loop thread.

3.1.2 ClEoProtoListT

```
typedef struct {
        ClUint8T protoID;
        ClInt8T name[20];
        ClEoProtoCallbackT func;
}ClEoProtoListT;
```

The structure, `ClEoProtoListT`, contains the list of protocols registered with EO. The attributes of this structure are:

- *protoID* - ID of the protocol being registered.
- *name* - Name of the protocol being registered.
- *func* - Receive function of the protocol.

3.1.3 ClEoSchedFeedBackT

typedef struct {
clEoPollingTypeT freq;
ClRcT status;
}ClEoSchedFeedBackT;

The structure, `ClEoSchedFeedBackT`, contains the feedback sent by the software component being polled in response to heartbeat, `(is-Alive)`. The attributes of this structure are:

- *freq* - Indicates the polling type `clEoPollingTypeT`.
- *status* - Indicates the health of the EO.

3.1.4 ClEoServiceObjT

typedef struct clEoServiceObj {
*void (*func)();*
*struct clEoServiceObj *pNextServObj;*
}ClEoServiceObjT;

The structure, `ClEoServiceObjT`, contains the EO service object. The attributes of this structure are:

- *void (*func)()* - Pointer to the client service function.
- **pNextServObj* - Pointer to the next service on the same service ID.

3.1.5 ClEoClientObjT

typedef struct {
ClEoServiceObjT funcs[CL_EO_MAX_NO_FUNC];
ClEoDataT data;
} ClEoClientObjT;

This structure, `ClEoClientObjT`, contains the pointer to the callback functions to be registered with EO, and the data specific to the client. The attributes of this structure are:

- *funcs[CL_EO_MAX_NO_FUNC]* - Pointer to EO functions.
- *data* - Data that is specific to the client

3.1.6 ClEoConfigT

typedef struct {
ClCharT EOname[CL_EO_MAX_NAME_LEN];
ClOsalThreadPriorityT pri;
ClUint32T noOfThreads;
ClIocPortT reqIocPort;
ClUint32T maxNoClients;

```
        ClEoApplicationTypeT appType;
        ClEoAppCreateCallbackT clEoCreateCallout;
        ClEoAppDeleteCallbackT clEoDeleteCallout;
        ClEoAppStateChgCallbackT clEoStateChgCallout;
        ClEoAppHealthCheckCallbackT clEoHealthCheckCallout;
        ClEoCustomActionT clEoCustomAction;
} ClEoConfigT;
```

The structure, `ClEoConfigT`, contains the configuration parameters related to the EO and is passed to the `clEoCreate` function.

- *EOname[CL_EO_MAX_NAME_LEN]* - EO name.
- *pri* - EO thread priority.
- *noOfThreads* - Number of RMD threads.
- *reqIocPort* - Requested IOC communication port.
- *maxNoClients* - Maximum number of EO clients.
- *appType* - Indicates if the application needs the main thread.
- *clEoCreateCallout* - Application function that is called from `main()` during the initialization process.
- *clEoDeleteCallout* - Application function that is called when EO needs to be terminated.
- *clEoStateChgCallout* - Application function that is called when EO enters suspended state.
- *clEoHealthCheckCallout* - Application function that is called when EO health check is performed by Component Manager.
- *clEoCustomAction* - Application function that is called when a Water Mark is reached.

3.1.7 ClEoStateT

```
typedef enum {
        CL_EO_STATE_INIT = 0x1,
        CL_EO_STATE_ACTIVE = 0x2,
        CL_EO_STATE_STDBY = 0x4,
        CL_EO_STATE_SUSPEND = 0x8,
        CL_EO_STATE_STOP = 0x10,
        CL_EO_STATE_KILL = 0x20,
        CL_EO_STATE_RESUME = 0x40,
        CL_EO_STATE_FAILED = 0x80
} ClEoStateT;
```

The values of the enumeration, `ClEoStateT`, contains the various states of EO.

- *CL_EO_STATE_INIT* - Initial state of the EO.
- *CL_EO_STATE_ACTIVE* - EO is in the active state.
- *CL_EO_STATE_STDBY* - EO is in the standby state.

- *CL_EO_STATE_SUSPEND* - EO is the suspended state.
- *CL_EO_STATE_STOP* - EO is in the stopped state.
- *CL_EO_STATE_KILL* - EO is in the killed state.
- *CL_EO_STATE_RESUME* - EO is resumed from the standby state.
- *CL_EO_STATE_FAILED* - EO is in the failed state.

3.1.8 ClEoApplicationTypeT

typedef enum {

textitCL_EO_USE_THREAD_FOR_RECV = CL_TRUE,

textitCL_EO_USE_THREAD_FOR_APP = CL_FALSE
}ClEoApplicationTypeT;

- *CL_EO_USE_THREAD_FOR_RECV* - If this is selected, the main thread is used to receive the RMD message. The main thread is not blocked in `ClEoAppCreateCallbackT` and returns immediately.
- *CL_EO_USE_THREAD_FOR_APP* - The main thread is allotted to the user-application. The main thread is blocked in `ClEoAppCreateCallbackT` or used by the application and returns only when the `ClEoAppDeleteCallbackT` is called.

3.1.9 clEoPollingTypeT

typedef enum {

textitCL_EO_DONT_POLL = 0,

textitCL_EO_BUSY_POLL = 1,

textitCL_EO_DEFAULT_POLL = 2
}clEoPollingTypeT;

The enumeration, `clEoPollingTypeT`,

- *CL_EO_DONT_POLL* - Component Manager stops the heartbeat of an EO if `CL_CPM_DONT_POLL` is received in response to the heartbeat.
- *CL_EO_BUSY_POLL* - Component Manager increases the heartbeat timeout to the maximum polling timeout. You can configure the maximum timeout while configuring the Component Manager.
- *CL_EO_DEFAULT_POLL* - Component Manager continues with the default heartbeat timeout. You can configure the default timeout while configuring the Component Manager.

3.1.10 ClEOServiceInstallOrderT

typedef enum {

textitCL_EO_ADD_TO_FRONT = 0,

textitCL_EO_ADD_TO_BACK = 1
}ClEOServiceInstallOrderT;

The enumeration, `ClEOServiceInstallOrderT`, is used while installing a client function

- *CL_EO_ADD_TO_FRONT* - Adds to the front of the list.
- *CL_EO_ADD_TO_BACK* - Adds to end of the list. This is used with the `clEoServiceValidate()` function.

3.1.11 ClEoDataT

typedef ClOsalTaskDataT ClEoDataT;

The type of the EO data.

3.1.12 ClIocPortT

typedef ClUint32T ClIocPortT;

The type of the identifier to the IOC communication port.

3.1.13 ClEoIdT

typedef ClUint16T ClEoIdT;

The type of the EO ID, assigned to an EO as part of the registration with the Component Manager.

3.1.14 ClEoPayloadWithReplyCallbackT

typedef ClRcT (ClEoPayloadWithReplyCallbackT) (*
CL_IN ClEoDataT data,
CL_IN ClBufferHandleT inMsgHandle,
CL_OUT ClBufferHandleT outMsgHandle);

RMD with payload (EO data) and pointer to the `reply` function. This is the generic function prototype definition for all RMD functions, installed on the EO client object.

- *data* - Data provided while invoking `clEoClientInstall()`.
- *inMsgHandle* - Received message over RMD.
- *outMsgHandle* - Reply message if any.

3.1.15 ClEoPayloadWithReplyCallbackT

*typedef ClRcT (*ClEoCallFuncCallbackT) (*
CL_IN ClEoPayloadWithReplyCallbackT func,
CL_IN ClEoDataT eoArg,
CL_IN ClBufferHandleT inMsgHdl,
CL_OUT ClBufferHandleT outMsgHdl);

Function callout definition required for the `clEoWalk()` function.

- *func* - Function that needs to be invoked.
- *eoArg* - Arguments that need to be passed.
- *inMsgHdl* - Request packet received including the protocol header.
- *outMsgHdl* - Data portion of the response to a protocol (PDU).

3.2 Library Life Cycle APIs

3.2.1 clEoLibInitialize

clEoLibInitialize

Synopsis:
Initializes the EO library.

Header File:
clEoConfigApi.h

Library Files:
libClEo

Syntax:

```
ClRcT clEoLibInitialize();
```

Parameters:
None

Return values:
CL_OK: The function executed successfully.

Description:
This function is used to initialize the EO library. It creates a list that contains the mapping of EO port to EO objects.

Related APIs:
clEoLibFinalize

3.2.2 clEoLibFinalize

clEoLibFinalize

Synopsis:
Frees the resources of the EO component acquired during initialization.

Header File:
clEoConfigApi.h

Library Files:
libClEo

Syntax:

```
ClRcT clEoLibFinalize();
```

normalsize

Parameters:
None

Return values:
CL_OK: The function executed successfully.

Description:
Frees the resources of the EO component acquired during initialization of the EO library.

Related APIs:
clEoLibFinalize

3.3 Functional APIs

3.3.1 clEoWalk

clEoWalk

Synopsis:
Performs a walk operation on the EO component.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoWalk(
                CL_IN ClEoExecutionObjT          *pThis,
                CL_IN ClUint32T                  func,
                CL_IN ClEoCallFuncCallbackT      pFuncCallout,
                CL_IN ClBufferHandleT     inMsgHdl,
                CL_OUT ClBufferHandleT    outMsgHdl);
```

normalsize

Parameters:

pThis: Handle of the EO.

func: Function number of the function to be executed.

pFuncCallout: Function that performs the execution.

inMsgHdl: Request message received including protocol header.

outMsgHdl: (out) Data part of the response to a protocol (PDU).

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pThis` contains a NULL pointer.

CL_EO_ERR_FUNC_NOT_REGISTERED: This function is not registered.

CL_EO_ERR_EO_SUSPENDED: EO is in the suspended state.

Description:
This function is used to perform a walk operation through the EO for a given RMD function number. It calls `rmdInvoke` for every callback function registered with an EO for that RMD function number.

Related APIs:
clEoServiceValidate

3.3.2 clEoServiceValidate

clEoServiceValidate

Synopsis:
Validates the function registration.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoServiceValidate(
                CL_IN    ClEoExecutionObjT   *pThis,
                CL_IN    ClUint32T           func);
```

normalsize

Parameters:

pThis: Handle of the EO.

func: Function to be invoked.

Return values:

CL_OK: The function executed successfully.

CL_EO_ERR_FUNC_NOT_REGISTERED: The function is not registered.

CL_EO_ERR_EO_SUSPENDED: The EO is in the suspended state.

CL_ERR_NULL_POINTER: `pThis` contains a NULL pointer.

Description:
This function is used to validate if the function for which the request is made is registered. It can be used to check if the services provided by an EO is available before calling the `clEoWalk()` function.

Related APIs:
clEoWalk

3.3.3 clEoClientInstall

clEoClientInstall

Synopsis:
Installs the function table for a client.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoClientInstall(
                CL_IN ClEoExecutionObjT      *pThis,
                CL_IN ClUint32T              clientID,
                CL_IN ClEoPayloadWithReplyCallbackT *pFuncs,
                CL_IN ClEoDataT              data,
                CL_IN ClUint32T              nFuncs);
```

normalsize

Parameters:

pThis: Handle of the EO.

clientId: ID of the client.

pFuncs: Pointer to the function table.

data: Data specific to the client.

nFuncs: Number of functions that are being installed.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pThis` or `pFuncs` contains a NULL pointer.

CL_EO_NO_MEMORY: Memory allocation failure.

CL_EO_CL_INVALID_CLIENTID: The client ID is invalid.

CL_EO_CL_INVALID_SERVICEID: The service ID is invalid.

Description:
This function is called by the client application to install its function table with the EO. The client exports the APIs that it provides to the users using this function. This function can be invoked through RMD calls.

Related APIs:
clEoClientUninstall

3.3.4 clEoClientUninstall

clEoClientUninstall

Synopsis:
Uninstalls the function table for the client.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoClientUninstall(
                CL_IN ClEoExecutionObjT* pThis,
                CL_IN ClUint32T clientId);
```

normalsize

Parameters:
pThis: Handle of the EO.

clientId: ID of the client.

Return values:
CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pThis` contains a NULL pointer.

CL_EO_ERR_INVALID_CLIENTID: The clientID is invalid.

Description:
This function is called by the client to uninstall its function table with the EO. After this function is successfully executed, the functions previously exported by this client using `clEoClientInstall()`, cannot be invoked as RMD calls.

Related APIs:
clEoClientInstall

3.3.5 clEoClientDataSet

clEoClientDataSet

Synopsis:
Stores the data specific to the client.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoClientDataSet(
             CL_IN ClEoExecutionObjT  *pThis,
             CL_IN ClUint32T          clientId,
             CL_IN ClEoDataT          data);
```

normalsize

Parameters:

pThis: Handle of the EO.

clientId: ID of the client.

data: Data specific to the client.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pThis` contains a NULL pointer.

CL_EO_ERR_INVALID_CLIENTID: The client ID is invalid.

Description:
This function is used to store the data specific to the client application.

Related APIs:
clEoClientDataGet

3.3.6 clEoClientDataGet

clEoClientDataGet

Synopsis:
Returns the client specific data.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoClientDataGet(
             CL_IN ClEoExecutionObjT *pThis,
             CL_IN   ClUint32T   clientId,
             CL_OUT  ClEoDataT   *pData);
```

normalsize

Parameters:

pThis: Handle of the EO.

clientId: ID of the client.

pData: (out) Data specific to the client.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pThis` or `pData` contains a NULL pointer.

CL_EO_ERR_INVALID_CLIENTID: The client ID is invalid.

Description:
This function is used to retrieve the data specific to the client .

Related APIs:
clEoClientDataSet

3.3.7 clEoServiceInstall

clEoServiceInstall

Synopsis:
Installs a particular client function.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoServiceInstall(
                CL_IN ClEoExecutionObjT              *pThis,
                CL_IN ClEoPayloadWithReplyCallbackT  pFunction,
                CL_IN ClUint32T                      iFuncNum,
                CL_IN ClUint32T                      order);
```

normalsize

Parameters:

pThis: Handle of the EO.

pFunction: Function pointer to be installed.

iFuncNum: Function number.

order: Specifies if the service that is to be installed, should be added to the front or the end of the table.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pThis` contains a NULL pointer.

CL_ERR_NO_MEMORY: Memory allocation failure.

CL_EO_CL_INVALID_SERVICEID: The service ID is invalid.

CL_ERR_INVALID_PARAMETER: A parameter is not set correctly.

Description:
This function is used to install a particular client function, identified by `iFuncNum` in the EO function table. Using this function, the application can register the service it provides to other components. This function can install the new service either to the front or back of the table by specifying the `order`.

Related APIs:
clEoServiceUninstall

3.3.8 clEoServiceUninstall

clEoServiceUninstall

Synopsis:
Uninstalls a particular client function.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoServiceUninstall(
            CL_IN ClEoExecutionObjT* pThis,
            CL_IN ClEoPayloadWithReplyCallbackT pFunction,
            CL_IN ClUint32T iFuncNum);
```

normalsize

Parameters:

pThis: Handle of the EO.

pFunction: Function pointer to be uninstalled.

iFuncNum: Function number.

Return values:

CL_OK: The function executed successfully.

CL_ERR_NULL_POINTER: `pThis` contains a NULL pointer.

CL_EO_FUNC_NOT_REGISTERED: Cannot de-register a function that is not registered.

CL_EO_CL_INVALID_SERVICEID: The service ID is invalid.

CL_ERR_INVALID_PARAMETER: A parameter is not set correctly.

Description:
This function is used to uninstall a particular client function from the EO function table. After this function is executed, the service, *pFunction*, becomes unavailable to be invoked as an RMD call.

Related APIs:
clEoServiceInstall

3.3.9 clEoPrivateDataSet

clEoPrivateDataSet

Synopsis:
Stores data in the area specific to the EO.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoPrivateDataSet(
                CL_IN ClEoExecutionObjT* pThis,
                CL_IN ClUint32T type,
                CL_IN void *pData);
```

normalsize

Parameters:

pThis: Handle of the EO.

type: User specified key.

pData: EO specific data.

Return Values:

CL_ERR_NULL_POINTER: `pThis` or `pDdata` contains a NULL pointer.

This function also returns the return values of the `clCntNodeAdd()` function.

Description:
This function is used to store data in a data area specific to EO. For a unique key, there can be only one node.

Related APIs:
clEoPrivateDataGet

3.3.10 clEoPrivateDataGet

clEoPrivateDataGet

Synopsis:
Retrieves data from the area specific to the EO.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoPrivateDataGet(
                CL_IN ClEoExecutionObjT* pThis,
                CL_IN ClUint32T type,
                CL_OUT void **data);
```

normalsize

Parameters:

pThis: Handle of the EO.

type: User specified key.

data: (out) Data specific to the EO.

Return values:

CL_ERR_NULL_POINTER: `pThis` or `data` contains a NULL pointer.

This function also returns the result of `clCntNodeUserDataGet`.

Description:
This function is used to retrieve the data private to the EO.

Related APIs:
clEoMyEoIocPortSet

3.3.11 clEoMyEoIocPortSet

clEoMyEoIocPortSet

Synopsis:
Sets the EO thread `iocPort`.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoMyEoIocPortSet(
                CL_IN ClIocPortT iocPort);
```

normalsize

Parameters:
iocPort: Contains the IOC port information that needs to be set.

Return values:
CL_OK: The function executed successfully.

Description:
This function is used to set the EO ID.

Related APIs:
clEoMyEoIocPortGet

3.3.12 clEoMyEoIocPortGet

clEoMyEoIocPortGet

Synopsis:
Retrieves the `IocPort` information of the EO.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoMyEoIocPortGet(
                CL_OUT ClIocPortT *pIocPort);
```

normalsize

Parameters:
plocPort: (out) This is populated with the retrieved IOC port.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_STATE: The state of the EO is invalid.

CL_ERR_NULL_POINTER: `pIocPort` contains a NULL pointer.

Description:
This function is used to retrieve the EO `IocPort`.

Related APIs:
clEoMyEoIocPortSet

3.3.13 clEoMyEoObjectSet

clEoMyEoObjectSet

Synopsis:
Stores the EO Object.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoMyEoObjectSet(
             CL_IN ClEoExecutionObjT* eoObj);
```

normalsize

Parameters:
pEoObj: Contains the `ClEoExecutionObjT` information that is to be stored.

Return values:
CL_OK: The function executed successfully.

Description:
This function is used to store the EO Object.

Related APIs:
clEoMyEoObjectGet.

3.3.14 clEoMyEoObjectGet

clEoMyEoObjectGet

Synopsis:
Retrieves the EO Object.

Header File:
clEoApi.h

Library Files:
ClEo

Syntax:

```
extern ClRcT clEoMyEoObjectGet(
             CL_OUT ClEoExecutionObjT** pEOObj);
```

normalsize

Parameters:
pEOObj: (out) This is populated with the EO object.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_STATE: The state of the EO is invalid.

Description:
This function is used to retrieve the EO Object.

Related APIs:
clEoMyEoObjectSet.

Chapter 4

Service Management Information Model

TBD

Chapter 5

Service Notifications

TBD

Chapter 6

Configuration

TBD

Chapter 7

Debug CLI

TBD

Index