OpenClovis

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

For OpenClovis SDK Release2.3 V0.3

Document Revision Date: December 19, 2006

Note: This document is not subject of the GPL license, even if you have obtained this document as a part of the GPL-ed version of OpenClovis SDK.

Contents

Chapter 1

Functional Overview

The OpenClovis Availability Management Framework (AMF) is built with the close association of two OpenClovis ASP components: Component Manager and Availability Management Service (AMS). These two components work together as defined by the Service Availability Forum. AMS is the primary decision making entity in the system with regard to the system model. It describes the various software and hardware components, attributes, inter-dependencies, service grouping rules and policies to apply in the event of failure. The CPM is the operational manager entity and is responsible for actually managing the various life cycles, based on configured policy and directions from the AMS.

AMF is a software entity that provides a framework for high availability of applications in a system. It is responsible for instantiating and managing all the OpenClovis ASP services and application components in a system. AMF executes configured recovery actions on the failure of application components. It also escalates the faults to larger scope as per the pre-configured escalation rules.

Functions of AMF

AMF maintains the view of one logical cluster that comprises several cluster nodes. These nodes host various resources in a distributed computing environment that describes the various software and hardware components of the system.

AMF stores information about attributes of the resources, their dependencies, rules for grouping them together for providing services, their current state, and the set of policies to be applied during failures. Using OpenClovis IDE, you can configure the desired availability policies with AMF on how to recover in case of failure of a service.

Availability Management Framework Functions

The Availability Management Framework administrative operations are applicable to the entities that are controlled by AMF, such as the components and CSIs. Therefore, restarting a node using an AMF administration function restarts all the components contained in the SUs housed in the node. This operation does not physically reboot the node because rebooting the physical node is outside the scope of AMF domain.

Most AMF administrative operations are applicable to the Service Unit (SU) logical entity and to the entities to which it belongs, such as Service Group (SG) or a Node. In certain cases, an exception has been made where the administration operation directly affects a component within an SU but those exceptions are rare.

The general look and feel of these functions is of the form clAmsEntityOP() where OP denotes the administrative operation such as lock, unlock, terminate etc. and the Entity represents the AMF logical entity on which this procedure is applicable.

Chapter 2

Service APIs

2.1 Type Definitions

2.1.1 ClAmsMgmtHandleT

typedef ClHandleT ClAmsMgmtHandleT;

The type of the handle of various AMS APIs.

2.1.2 ClAmsMgmtCallbacksT

typedef struct {
int nothingForNow
} ClAmsMgmtCallbacksT;

2.2 Library Life Cycle APIs

2.2.1 clAmsMgmtInitialize

clAmsMgmtInitialize

Synopsis:

Starts the use of the management function library.

Header File:

clAmsMgmtClientApi.h

Syntax:

```
    ClRcT clAmsMgmtInitialize(
                            CL_OUT      ClAmsMgmtHandleT            *amsHandle,
                            CL_IN       const ClAmsMgmtCallbacksT   *amsMgmtCallbacks,
                            CL_INOUT    ClVersionT                  *version);
```

Parameters:

amsHandle: (out) The handle returned by the function. It identifies a particular initialization of the Availability Management Framework. This handle must be passed as the first input parameter for all further usage of the function library.

amsMgmtCallbacks: (in)Callbacks into management function user.

version: (in/out) In the input parameter, you must pass the current version of AMS on the client. In the output parameter, you will receive the supported version.

Return values:

CL_OK: The function executed successfully.

CL_ERR_VERSION_MISMATCH: If the current version and the supported version are not the same.

Description:

This function is used to start the use of the management function library. It typically registers callbacks and it must be called before invoking any other function of the AMS library. In this release there are no callbacks to be registered.

Library File:

ClAmsMgmtClient

Related Function(s):

clAmsMgmtFinalize

2.2.2 clAmsMgmtFinalize

clAmsMgmtFinalize

Synopsis:

Terminates the use of the management function library.

Header File:

clAmsMgmtClientApi.h

Syntax:

```
    ClRcT clAmsMgmtFinalize(
                                  CL_IN       ClAmsMgmtHandleT           amsHandle);
```

Parameters:

amsHandle: (in)The handle, obtained through the *clAmsMgmtInitialize()* function, designating a particular initialization of the Availability Management Framework.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: *amsHandle* was invalid. This handle must be same as that obtained through the clAmsMgmtInitialize function.

Description:

This function is used to terminate the use of the management function library. This must be called when the services of the AMS library are no longer required. This function frees all resources allocated during initialization of the library through the *clAmsMgmtInitialize()* function.

Library File:

ClAmsMgmtClient

Related Function(s):

clAmsMgmtInitialize

2.3 Functional APIs

2.3.1 clAmsMgmtEntityLockAssignment

clAmsMgmtEntityLockAssignment

Synopsis:

Prevents an AMS entity from taking work.

Header File:

clAmsMgmtClientApi.h

Syntax:

```
ClRcT clAmsMgmtEntityLockAssignment(
                             CL_IN       ClAmsMgmtHandleT            amsHandle,
                            CL_IN       const ClAmsEntityT          *entity);
```

Parameters:

amsHandle: (in)The handle, obtained through the *clAmsMgmtInitialize()* function, designating a particular initialization of the Availability Management Framework.

entity: (in) Entity type and name: Attributes of the entity that is to be prevented from taking work.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: Handle was invalid.

CL_ERR_NOT_EXIST: Resource (Entity) does not exist.

Description:

This function is used to prevent an AMS entity from taking work. This administrative operation is applicable to all Availability Management Framework entities that possess an administrative state, namely, service unit, service instance, node, and service group.
The invocation of this function sets the administrative state of the logical entity designated by *entityName* to `locked`. Further transition to lock instantiate or unlock administrator state can happen only through lock assignment administrator state.
The entity is identified either by a handle, or by the entity type and name if it is NULL.

Library File:

ClAmsMgmtClient

Related Function(s):

clAmsMgmtEntityLockInstantiation, clAmsMgmtEntityUnlock

2.3.2 clAmsMgmtEntityLockInstantiation

clAmsMgmtEntityLockInstantiation

Synopsis:

Prevents an AMS entity from being instantiated.

Header File:

clAmsMgmtClientApi.h

Syntax:

```
    ClRcT clAmsMgmtEntityLockInstantiation(
                                CL_IN       ClAmsMgmtHandleT             amsHandle,
                                CL_IN       const ClAmsEntityT           *entity);
```

Parameters:

amsHandle: (in)The handle, obtained through the *clAmsMgmtInitialize()* function, designating a particular initialization of the Availability Management Framework.

entity: (in)Entity type and name.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: *amsHandle* was invalid. This handle must be same as that obtained through the *clAmsMgmtInitialize()* function.

CL_ERR_INVALID_STATE: Error: The operation is not valid in the current state.

CL_ERR_NOT_EXIST: Resource (Entity) does not exist.

Description:

This function is used to prevent an AMS entity from being instantiated. This administrative operation is applicable to all Availability Management Framework entities that possess an administrative state namely, service unit, service instance, node, and service group.
Transition to lock instantiation administrative state can happen only through lock assignment administrative state. Therefore any entity which needs to be assigned instantiation lock must be in the lock-assigned state.
The entity is identified either by a handle, or by the entity type and name if it is NULL.

Library File:

ClAmsMgmtClient

Related Function(s):

clAmsMgmtEntityLockAssignment, clAmsMgmtEntityUnlock

2.3.3 clAmsMgmtEntityUnlock

clAmsMgmtEntityUnlock

Synopsis:
Unlocks an AMS entity for utilization by AMS.

Header File:
clAmsMgmtClientApi.h

Syntax:

```
    ClRcT clAmsMgmtEntityUnlock(
                                        CL_IN       ClAmsMgmtHandleT           amsHandle,
                                        CL_IN       const ClAmsEntityT         *entity);
```

Parameters:

amsHandle: (in)The handle, obtained through the *clAmsMgmtInitialize()* function, designating a particular initialization of the Availability Management Framework.

entity: (in)Entity type and name.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: *amsHandle* was invalid. This handle must be same as that obtained through the *clAmsMgmtInitialize()* function.

CL_ERR_INVALID_STATE: The operation is not valid in the current state.

CL_ERR_NOT_EXIST: Resource (Entity) does not exist.

Description:
This function is used to unlock an AMS Management entity, to enable it to be utilized by AMS. This administrative operation is applicable to all Availability Management Framework entities that possess an administrative state, namely, service unit, service instance, node and service group.
The invocation of this function sets the administrative state of the logical entity designated by *entityName* to `unlocked`. Transition to the unlock administrator state can happen only through lock assignment administrator state. So any entity that needs to be unlocked must be in the lock assigned state.If this operation is invoked on an entity that is already unlocked, there is no change in the status of such an entity and it remains unlocked.

Library File:
ClAmsMgmtClient

Related Function(s):
clAmsMgmtEntityLockAssignment, clAmsMgmtEntityLockInstantiation

2.3.4 clAmsMgmtEntityShutdown

clAmsMgmtEntityShutdown

Synopsis:
Shuts down an AMS entity.

Header File:
clAmsMgmtClientApi.h

Syntax:

```
    ClRcT clAmsMgmtEntityShutdown(
                                CL_IN      ClAmsMgmtHandleT          amsHandle,
                                CL_IN      const ClAmsEntityT        *entity);
```

Parameters:

amsHandle: (in)The handle, obtained through the *clAmsMgmtInitialize()* function, designating a particular initialization of the Availability Management Framework.

entity: (in)A pointer to the name of the logical entity to be terminated.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: *amsHandle* was invalid. This handle must be same as that obtained through the *clAmsMgmtInitialize()* function.

CL_ERR_INVALID_STATE: The operation is not valid in the current state.

CL_ERR_NOT_EXIST: Resource/Entity does not exist.

Description:
This function is used to shutdown an AMS entity. This administrative operation is applicable to service unit, service instance, cluster node and service group i.e. all Availability Management Framework entities that support an administrative state.
The invocation of this function sets the administrative state of the logical entity designated by *entityName* to shutting down. This administrative operation is non-blocking i.e. it does not wait for the logical entity designated by *entityName* to transition to locked state which can take a long time.
If this operation is invoked on an entity that is already shutting-down or locked there is no change in the status of such an entity i.e. it continues shutting-down or remains locked in such a case.
The entity is identified either by a handle, or if it is NULL, then by the entity type and name.

Library File:
ClAmsMgmtClient

Related Function(s):
clAmsMgmtEntityRepaired, clAmsMgmtEntityRestart

2.3.5 clAmsMgmtEntityRestart

clAmsMgmtEntityRestart

Synopsis:
Restarts an AMS entity.

Header File:
clAmsMgmtClientApi.h

Syntax:

```
ClRcT clAmsMgmtEntityRestart(
                              CL_IN     ClAmsMgmtHandleT     amsHandle,
                              CL_IN     const ClAmsEntityT   *entity);
```

Parameters:

amfHandle: (in)The handle, obtained through the *clAmsMgmtInitialize()* function, designating a particular initialization of the Availability Management Framework.

entity: (in)Entity type and name.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: *amsHandle* was invalid. This handle must be same as that obtained through the *clAmsMgmtInitialize()* function.

CL_ERR_INVALID_STATE: The operation is not valid in the current state.

CL_ERR_NOT_EXIST: Resource/Entity does not exist.

Description:
This function is used to restart an AMS entity. This administrative operation is applicable to service units whose present state is instantiated. The invocation of this function on a service unit causes the SU to be restarted by restarting all the components within it.
If there are any SIs or CSIs that are affected by this action, they are recovered as per the recovery policy for the restarted entity. This function is valid for node, SU, application and component. Note that this is one of the few administrative operations that are applicable to a component.If the recovery policy for all components in an SU is to restart and it is not disabled for any component, then no service assignments to other entities are required.
An entity is identified either by a handle, or by the entity type and name if it is null.

Library File:
ClAmsMgmtClient

Related Function(s):
clAmsMgmtEntityRepaired, clAmsMgmtEntityShutdown

2.3.6 clAmsMgmtEntityRepaired

clAmsMgmtEntityRepaired

Synopsis:

Indicates that a faulty AMS entity has been repaired.

Header File:

clAmsMgmtClientApi.h

Syntax:

```
        ClRcT clAmsMgmtEntityRepaired(
                                    CL_IN       ClAmsMgmtHandleT           amsHandle,
                                    CL_IN       const ClAmsEntityT         *entity);
```

Parameters:

amfHandle: (in)The handle, obtained through the *clAmsMgmtInitialize()* function, designating a particular initialization of the Availability Management Framework.

entity: (in)Entity type and name.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: *amsHandle* was invalid. This handle must be same as that obtained through the *clAmsMgmtInitialize()* function.

CL_ERR_INVALID_STATE: The operation is not valid in the current state.

CL_ERR_NOT_EXIST: Resource (Entity) does not exist.

Description:

This function is used to indicate that a faulty AMS entity has been repaired and can be put back into service. This function is only valid for node and SU.
The entity is identified either by a handle, or by the entity type and name if its null.

Library File:

ClAmsMgmtClient

Related Function(s):

clAmsMgmtEntityRestart, clAmsMgmtEntityShutdown

2.3.7 clAmsMgmtDebugEnable

clAmsMgmtDebugEnable

Synopsis:

Enables debugging for the given entity.

Header File:

clAmsMgmtClientApi.h

Syntax:

```
    ClRcT clAmsMgmtDebugEnable(
                              CL_IN     ClAmsMgmtHandleT        amsHandle,
                              CL_IN     const ClAmsEntityT      *entity,
                              CL_IN     ClUint8T                debugFlags );
```

Parameters:

amfHandle: (in)The handle, obtained through the *clAmsMgmtInitialize()* function, designating a particular initialization of the Availability Management Framework.

entity: (in)Entity type and name for which debugging needs to be enabled. If this entity is NULL, then it is assumed that the debug operation is to be performed for all entities in AMS.

debugFlags: (in)Specific to AMS, this parameter defines the section of the code for which debugging needs to be enabled.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: *amsHandle* was invalid. This handle must be same as that obtained through the *clAmsMgmtInitialize()* function.

CL_ERR_INVALID_STATE: The operation is not valid in the current state.

CL_ERR_NOT_EXIST: Resource (Entity) does not exist.

Description:

Debugging can be enabled for all entities in AMS or for a particular entity. The debug flags indicate the type of messages which are currently enabled for the entity or entire AMS. The various debug flags are:

- msg: Disables AMF messages for important AMF events such as work assignment, admin operation, and so on.
- timer: Disables AMF messages for timer-related AMF events such as timer start, timer stop, and timer expiration.
- state_change: Disables AMF messages for state change related events such as component changing its present state from uninstantiated to instantiated.
- function_enter: Disables function enter messages for AMS. This is useful for debugging the stack trace of functions called by AMF during its operation.
- all: Disables all of above debug flags for AMS. *all* is the equivalent of msg|timer|state_change|function_enter.

debugFlags is a combination of the above kinds of messages.

Library File:

ClAmsMgmtClient

Related Function(s):

clAmsMgmtDebugDisable , clAmsMgmtDebugGet

2.3.8 clAmsMgmtDebugDisable

clAmsMgmtDebugDisable

Synopsis:
Disables debugging for the given entity.

Header File:
clAmsMgmtClientApi.h

Syntax:

```
ClRcT clAmsMgmtDebugDisable(
                              CL_IN ClAmsMgmtHandleT amsHandle,
                              CL_IN const ClAmsEntityT *entity,
                              CL_IN ClUint8T debugFlags );
```

Parameters:

amfHandle: (in)The handle, obtained through the *clAmsMgmtInitialize()* function, designating a particular initialization of the Availability Management Framework.

entity: (in)Entity type and name for which debugging needs to be enabled.

debugFlags: (in) Specific to AMS, this parameter defines the section of the code for which the debugging needs to be disabled.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: *amsHandle* was invalid. This handle must be same as that obtained through the *clAmsMgmtInitialize()* function.

CL_ERR_INVALID_STATE: The operation is not valid in the current state,

CL_ERR_NOT_EXIST: Resource (Entity) does not exist.

Description:
This function is used to return debugging flags for an AMS entity or for entire AMS. The debug flags indicate the type of messages that are currently enabled for the entity or entire AMS. The various debug flags are:

- msg: Disables AMF messages for important AMF events such as work assignment, administrator operation, and so on.
- timer: Disables AMF messages for timer related AMF events such as timer start, timer stop, and timer expiration.
- state_change: Disables AMF messages for events related to state change, such as a component changing its presence state from uninstantiated to instantiated.
- function_enter: Disables function enter messages for AMS. This is useful for debugging the stack trace of functions called by AMF during its operation.
- all: Disables all of above debug flags for AMS. *all* is the equivalent of msg|timer|state_change|function_enter.

Library File:
ClAmsMgmtClient

Related Function(s):
clAmsMgmtDebugEnable clAmsMgmtDebugGet

2.3.9 clAmsMgmtDebugGet

clAmsMgmtDebugGet

Synopsis:
Returns debugging information for the given entity.

Header File:
clAmsMgmtClientApi.h

Syntax:

```
ClRcT clAmsMgmtDebugGet(
                              CL_IN ClAmsMgmtHandleT  amsHandle,
                              CL_IN const ClAmsEntityT *entity,
                              CL_OUT ClUint8T *debugFlags );
```

Parameters:

amfHandle: (in) The handle, obtained through the *clAmsMgmtInitialize()* function, designating a particular initialization of the Availability Management Framework.

entity: (in) Entity type and name for which debugging information is requested.

debugFlags: (out) Specific to AMS, this parameter defines the section of the code for which the debugging information is to be retrieved.

Return values:

CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: *amsHandle* was invalid. The handle passed to this function must be the same as the one obtained from the *clAmsMgmtInitialize()* function.

CL_ERR_INVALID_STATE: The operation is not valid in the current state.

CL_ERR_NOT_EXIST: Resource (Entity) does not exist.

Description:
This function is used to return debugging flags for an AMS entity or for entire AMS. The debug flags indicate the type of messages which are currently enabled for the entity or entire AMS. The various debug flags are:

- msg: Disables AMF messages for important AMF events such as work assignment, admin operation, and so on.
- timer: Disables AMF messages for timer related AMF events such as timer start, timer stop, and timer expiration.
- state_change: Disables AMF messages for state change related events such as component changing its presence state from uninstantiated to instantiated.
- function_enter: Disables function enter messages for AMS. This is useful for debugging the stack trace of functions called by AMF during its operation.
- all: Disables all of above debug flags for AMS. *all* is the equivalent of msg|timer|state_change|function_enter.

Library File:
ClAmsMgmtClient

Related Function(s):
clAmsMgmtDebugDisable , clAmsMgmtDebugEnable

2.3.10 clAmsMgmtDebugEnableLogToConsole

clAmsMgmtDebugEnableLogToConsole

Synopsis:
Enables AMS debugging messages to be displayed on the console.

Header File:
clAmsMgmtClientApi.h

Syntax:

```
ClRcT clAmsMgmtDebugEnableLogToConsole(
                                CL_IN ClAmsMgmtHandleT amsHandle );
```

Parameters:
amfHandle: (in) Handle assigned to API user.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: Error: Handle was invalid.

Description:
This function is used to return debugging flags for an AMS entity or for entire AMS. The debug flags indicate the type of messages which are currently enabled for the entity or entire AMS. The various debug flags are:

- msg: Disables AMF messages for important AMF events such as work assignment, admin operation, and so on.
- timer: Disables AMF messages for timer related AMF events such as timer start, timer stop, and timer expiration.
- state_change: Disables AMF messages for state change related events such as component changing its presence state from uninstantiated to instantiated.
- function_enter: Disables function enter messages for AMS. This is useful for debugging the stack trace of functions called by AMF during its operation.
- all: Disables all of above debug flags for AMS. *all* is the equivalent of msg|timer|state_change|function_enter.

Library File:
ClAmsMgmtClient

Related Function(s):

2.3.11 clAmsMgmtDebugDisableLogToConsole

clAmsMgmtDebugDisableLogToConsole

Synopsis:
Disables AMS debugging messages to be displayed on the console.

Header File:
clAmsMgmtClientApi.h

Syntax:

```
    ClRcT clAmsMgmtDebugDisableLogToConsole(
                                      CL_IN ClAmsMgmtHandleT amsHandle );
```

Parameters:
amfHandle: (in) Handle assigned to API user.

Return values:
CL_OK: The function executed successfully.

CL_ERR_INVALID_HANDLE: Error: Handle was invalid.

Description:
This function is used to return debugging flags for an AMS entity or for entire AMS. The debug flags indicate the type of messages which are currently enabled for the entity or entire AMS. The various debug flags are:

- msg: Disables AMF messages for important AMF events such as work assignment, admin operation, and so on.
- timer: Disables AMF messages for timer related AMF events such as timer start, timer stop, and timer expiration.
- state_change: Disables AMF messages for state change related events such as component changing its presence state from uninstantiated to instantiated.
- function_enter: Disables function enter messages for AMS. This is useful for debugging the stack trace of functions called by AMF during its operation.
- all: Disables all of above debug flags for AMS. *all* is the equivalent of msg|timer|state_change|function_enter.

Library File:
ClAmsMgmtClient

Related Function(s):

Index