B Class RTI::FederateAmbassador

B.1 Federation Management

B.1.1 announceSynchronizationPoint()

RTI 1.3-NG

ABSTRACT

This service informs the federate that synchronization based on some federation-defined semantics has been requested

HLA IF SPECIFICATION

This method realizes the "Announce Synchronization Point" Federation Management service as specified in the *HLA Interface Specification* (§4.8 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   announceSynchronizationPoint (
      const char *label,
      const char *tag
   )
throw (
   RTI::FederateInternalError
)
```

ARGUMENTS

label

a string used to uniquely identify the synchronization point and convey federation-defined semantics

tag

the "user-supplied tag" passed to the invocation `registerFederationSynchronizationPoint()` used to create the synchronization point; this string is not interpreted by the RTI

DESCRIPTION

Synchronization points provide a mechanism for federates to schedule checkpoints with federation-defined semantics, while relying on the RTI to perform the bookkeeping associated with determining when the checkpoint is achieved by the desired set of federates.

This callback informs a federate that it has been requested to participate in a synchronization point. The synchronization point may be a *universal* synchronization point that applies to all federates or a *specified* synchronization point in which the federate has been explicitly included. The semantics of the synchronization point are defined by the federation and communicated through the associated label and user-supplied tag.

The federate should use the `synchronizationPointAchieved()` service to notify the federation when it has met the synchronization criteria. When all federates included in the synchronization point have achieved synchronization, each included federate will receive a `federationSynchronized()` callback.

Depending on the semantics of a synchronization point, it may or may not be appropriate for a federate to continue operation while waiting for synchronization to be achieved. The RTI places no restrictions on a federation pending synchronization; federation developers are free to implement their own restrictions based on federation-specific synchronization semantics. **At a minimum, all federates must continue to invoke `tick()` so that internal RTI communications may be serviced.**

For *specified* synchronization points, throwing an exception from this method will result in the synchronization point never being attained. For *universal* synchronization points, the synchronization point may be reannounced to the federate upon the subsequent joining of federates to the federation.

If a federate resigns before achieving an announced synchronization point, the federate is removed from consideration for that synchronization point (i.e., it is assumed to have met the synchronization criteria).

RETURN VALUES

A non-exceptional return indicates that the federate understands the synchronization request and will subsequently notify the RTI when the synchronization criteria have been met.

EXCEPTIONS

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador

federationSynchronized()

RTI::RTIambassador

registerFederationSynchronizationPoint()
resignFederationExecution()
synchronizationPointAchieved()

B.1.2 federationNotRestored()

RTI 1.3-NG

ABSTRACT

This service informs the federate that a federation-wide restoration attempt has completed, but that one or more federates have failed to correctly restore their state.

HLA IF SPECIFICATION

This method (in conjunction with `federationRestored()`) realizes the "Federation Restored" Federation Management service as specified in the *HLA Interface Specification* (§4.21 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   federationNotRestored ( )
throw (
   RTI::FederateInternalError
)
```

DESCRIPTION

This callback informs a federate of the fact that a federation-wide restoration has ended unsuccessfully (i.e., one or more of the restoring federates invoked the `federateRestoreNotComplete()` service to end the restoration, or one or more LRCs were unable to restore their internal state.) Only one restore may be in progress simultaneously, so the subject of this callback is always the most recent restoration attempt as announced via a `initiateFederateRestore()` callback.

The failure of a single federate or LRC does not preclude the successful restoration of other federates or LRCs, so the state of the federation after a failed restoration attempt is not well defined.

Upon receipt of such a notification, the federate is no longer in a suspended state and may resume normal operation.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the failure to restore federation state. The federate may resume normal operation.

EXCEPTIONS

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador

federationRestoreBegun()
federationRestored()
initiateFederateRestore()

RTI::RTIambassador

federateRestoreComplete()
federateRestoreNotComplete()
requestFederationRestore()

B.1.3 federationNotSaved()

RTI 1.3-NG

ABSTRACT

This service informs the federate that a federation-wide save attempt has completed, but that one or more federates have failed to correctly save their state.

HLA IF SPECIFICATION

This method (in conjunction with `federationSaved()`) realizes the "Federation Saved" Federation Management service as specified in the *HLA Interface Specification* (§4.15 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   federationNotSaved( )
throw (
   RTI::FederateInternalError
)
```

DESCRIPTION

This callback informs a federate of the fact that a federation-wide restoration has ended unsuccessfully. Only one restore may be in progress simultaneously, so the subject of this callback is always the most recent restoration attempt as announced via an `initiateFederateRestore()` callback.

The failure of a single federate or LRC does not preclude the successful restoration of other federates or LRCs, so the state of the federation after a failed restoration attempt is not well defined.

Upon receipt of such a notification, the federate is no longer in a suspended state and may resume normal operation.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the failure to save federation state. The federate may resume normal operation.

EXCEPTIONS

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador

federationSaved()
initiateFederateSave()

RTI::RTIambassador

federateSaveComplete()
federateSaveNotComplete()
requestFederationSave()

B.1.4 federationRestoreBegun()

RTI 1.3-NG

ABSTRACT

This service advises the federate that a federation-wide restoration has begun. The federate is suspended from making service invocations that would change the state of the federation until the restoration has completed.

HLA IF SPECIFICATION

This service realizes the "Federation Restore Begun" Federation Management service as specified in the *HLA Interface Specification* (§4.18 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   federationRestoreBegun ( )
throw (
   RTI::FederateInternalError
)
```

DESCRIPTION

This callback instructs a federate to refrain from making service invocations that would change the state of the local LRC or the federation, in anticipation of a pending restoration attempt. The federation-wide restoration then proceeds according to the following sequence of events:

1. When all federates have been suspended in this fashion, the LRCs comprising the federation will commence unmarshaling their internal state from disk.
2. When all LRCs have completed unmarshaling their respective states, each federate will be instructed to restore its federate-managed state via an `initiateFederateRestore()` callback.
3. When all federates have completed the restoration of their federate-managed state (as reported using the `federateRestoreComplete()` or `federateRestoreNotComplete()` services), each federate will receive a `federationRestored()` or `federationNotRestored()` callback. Upon receipt of such, a federate may resume normal operation.

The LRC will refrain from delivering events to the federate during the restoration process. Following the restoration, any events queued in the restored LRC state will be delivered to the federate at the appropriate time.

The federate must continue to call `tick()` so that internal RTI communications may continue to be serviced.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the restoration and will suspend normal operation pending further notification.

EXCEPTIONS

RTI::FederateInternalError
: An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador
: initiateFederateRestore()

RTI::RTIambassador
: requestFederationRestore()

B.1.5 federationRestored()

RTI 1.3-NG

ABSTRACT

This service informs a federate that the currently outstanding federation-wide restoration attempt has completed successfully.

HLA IF SPECIFICATION

This method (in conjunction with `federationNotRestored()`) realizes the "Federation Restored" Federation Management service as specified in the *HLA Interface Specification* (§4.21 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   federationRestored ( )
throw (
   RTI::FederateInternalError
)
```

DESCRIPTION

This callback informs a federate of the fact that a federation-wide restoration has ended successfully (i.e. all restoring federates invoked the `federateRestoreComplete()` service and all LRCs successfully restored their internal state.) Only one restore may be in progress simultaneously, so the subject of this callback is always the most recent restoration attempt as announced via an `initiateFederateRestore()` callback.

Upon receipt of such a notification, the federate is no longer in a suspended state and may resume normal operation.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the successful completion of the federation-wide restoration. The federate may resume normal operation.

EXCEPTIONS

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador

federationRestoreBegun()
federationNotRestored()
initiateFederateRestore()

RTI::RTIambassador

federateRestoreComplete()
federateRestoreNotComplete()
requestFederationRestore()

B.1.6 federationSaved()

RTI 1.3-NG

ABSTRACT

This service informs a federate that the currently outstanding federation-wide save attempt has completed successfully.

HLA IF SPECIFICATION

This method (in conjunction with `federationNotSaved()`) realizes the "Federation Saved" Federation Management service as specified in the *HLA Interface Specification* (§4.15 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   federationSaved ( )
throw (
   RTI::FederateInternalError
)
```

DESCRIPTION

This callback informs a federate of the fact that a federation-wide save has ended successfully (i.e. all saving federates invoked the `federateSaveComplete()` service and all LRCs successfully save their internal state.) Only one save may be in progress simultaneously, so the subject of this callback is always the most recent save attempt as announced via an `initiateFederateSave()` callback.

Upon receipt of such a notification, the federate is no longer in a suspended state and may resume normal operation.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the successful completion of the federation-wide save. The federate may resume normal operation.

EXCEPTIONS

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador

federationNotSaved()
initiateFederateSave()

RTI::RTIambassador

federateSaveComplete()
federateSaveNotComplete()
requestFederationSave()

B.1.7 federationSynchronized()

RTI 1.3-NG

ABSTRACT

This service informs a federate that a synchronization point previously announced to the federate has been achieved by all relevant federates.

HLA IF SPECIFICATION

This method realizes the "Federation Synchronized" Federation Management service as specified in the *HLA Interface Specification* (§4.10 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   federationSynchronized (
      const char *label
   )
throw (
   RTI::FederateInternalError
)
```

ARGUMENTS

label

string label used to distinguish among simultaneously active synchronization points

DESCRIPTION

Synchronization points are a generalization of the pause/resume capabilities featured in early revisions of HLA. They provide a mechanism for federates to schedule checkpoints with federation-defined semantics, while relying on the RTI to perform the bookkeeping associated with determining when the checkpoint is achieved by the desired set of federates.

This callback notifies a federate that a synchronization point has been achieved, i.e.:

- For a *universal* synchronization point, all active federates in the federation (including recently joined federates) have achieved the synchronization point.
- For a *specified* synchronization point, all federates explicitly included in the synchronization point at registration have resigned or achieved the synchronization point.

The synchronization label should correspond to a synchronization point that has been previously announced to the federate through an `announceSynchronizationPoint()` callback and attained by the federate, as reported using `synchronizationPointAchieved()` service.

The semantics of the synchronization are defined by the federation and communicated through the synchronization-point label and associated user-supplied tag.

After federation-wide attainment of a synchronization point, the synchronization-point label may be reused.

RETURN VALUES

A non-exceptional return indicates that the federate recognizes the synchronization-point label and acknowledges the synchronization.

EXCEPTIONS

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador
announceSynchronizationPoint()
RTI::RTIambassador
registerFederationSynchronizationPoint()
synchronizationPointAchieved()

B.1.8 initiateFederateRestore()

RTI 1.3-NG

ABSTRACT

This service instructs a federate to restore its federate-managed state from the saved state associated with a specified label and federate handle.

HLA IF SPECIFICATION

This service realizes the "Initiate Federate Restore" Federation Management service as specified in the *HLA Interface Specification* (§4.19 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   initiateFederateRestore (
       const char *label,
       RTI::FederateHandle handle
   )
throw (
   RTI::SpecifiedSaveLabelDoesNotExist,
   RTI::CouldNotRestore,
   RTI::FederateInternalError
)
```

ARGUMENTS

label

a string label associated with a previously saved named federation state

handle

the new handle associated with the federate after restoration; this handle should correspond to a saved state for a federate of the same type as the current federate

DESCRIPTION

This callback instructs the federate to begin restoring its federate-managed state from a previously saved state associated with the specified label and federate handle. The federate will have been previously suspended via a `federationRestoreBegun()` callback and will remain suspended until the federation-wide restoration has completed. Prior to the issuance of this callback, the LRC associated with the federate will have restored its internal state based on the same save label and federate handle.

The federate handle will correspond to a federate in the saved federation of the same type as the active federate, as distinguished by the "name" argument to `joinFederationExecution()`. These names should be chosen in such a way as to ensure that a federate application will function correctly when restored to a saved LRC/federate state associated with any other federate of the same name.

When the federate has restored its federate-managed state or failed to do so, it should advise the RTI of such, using the `federateRestoreComplete()` or `federateRestoreNotComplete()` service, respectively. When all federates have completed the restoration of their federate-managed states, each federate will receive a `federationRestored()` or `federationNotRestored()` callback. Upon receipt of such, a federate may resume normal operation.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the restoration request and will subsequently notify the RTI when the restoration has been completed.

EXCEPTIONS

RTI::SpecifiedSaveLabelDoesNotExist

The specified save label does not correspond to an existing labeled saved state.

RTI::CouldNotRestore

The federate recognizes the save label but was unable to restore its state for some other reason.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
federationNotRestored()
federationRestoreBegun()
federationRestored()
RTI::RTIambassador::
getRegion()
requestFederationRestore()

B.1.9 initiateFederateSave()

RTI 1.3-NG

ABSTRACT

This service instructs the federate to save its state as of its current logical time.

HLA IF SPECIFICATION

This method realizes the "Initiate Federate Save" Federation Management service as specified in the *HLA Interface Specification* (§4.12 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   initiateFederateSave (
       const char *label
   )
throw (
   RTI::UnableToPerformSave,
   RTI::FederateInternalError
)
```

ARGUMENTS

label

string that was passed to invocation of the *RTIambassador::requestFederationSave* service that requested the save.

This parameter is not interpreted by the RTI itself. It provided as a means for the requesting a federate to specify a textual description of the reason for the save request or any other information meaningful in the context of the federation. The federate must make a copy of this parameter if it wishes to retain its value after the completion of the call.

DESCRIPTION

This callback instructs a federate to initiate a save of its federate-managed state as soon as possible. The federate's state should be saved to disk in such a way that it may be distinguished:

- from other federate-managed states with the same save label and a different federate handle
- from other federate-managed states with the same federate handle and a different save label

The federate is expected to begin saving its federate-managed state as soon as possible, and to inform the RTI of such using the `federateSaveBegun()` service. The federate should subsequently report success or failure of the save using the appropriate service.

The federate is suspended from invoking any services that would change the state of the LRC or the federation during the save attempt.

RETURN VALUES

A non-exceptional return value indicates that the federate has acknowledged the save initiation request and will begin saving its state as soon as possible, notifying the RTI of such using the `federateSaveBegun()` service.

NOTES

- If a logical time is associated with the save request, time-constrained federates will be instructed to save at the appropriate logical time. Logical time occurs when the set of time-stamp-ordered events that have been delivered to the federate exactly matches the set of relevant time-stamp-ordered events occurring in the federation with time-stamps less than or equal to the logical time of the save. All federates will be instructed to save only after all time-constrained federates have reached the logical time of the save.
- If no logical time is associated with the save request, the instruction to save will be delivered to each federate as soon as possible.
- The services used to report success or failure of a save are `federateSaveComplete()` and `federateSaveNotComplete()`, respectively. The later name is somewhat of a misnomer, as the method is actually used to indicate the completion of an unsuccessful save attempt.
- No events will be delivered to the federate while a save is in progress.
- The federate will remain suspended until the federation-wide save has been completed. The federate will be notified of such via a `federationSaved()` or `federationNotSaved()` callback.

EXCEPTIONS

RTI::UnableToPerformSave

The federate is unable to perform a save at the current time.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador
federationNotSaved()
federationSaved()
RTI::RTIambassador
federateSaveBegun()
federateSaveComplete()
federateSaveNotComplete()
requestFederationRestore()
requestFederationSave()

B.1.10 requestFederationRestoreFailed()

RTI 1.3-NG

ABSTRACT

This service informs a federate that a request to attempt to restore its federation state has been denied.

HLA IF SPECIFICATION

This method (in conjunction with `requestFederationRestoreSucceeded()`) realizes the "Confirm Federation Restoration Request" Federation Management service as specified in the *HLA Interface Specification* (§4.17 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   requestFederationRestoreFailed (
      const char *label,
      const char *reason)
throw (
   RTI::FederateInternalError
)
```

ARGUMENTS

label

the label associated with the restoration request

reason

a string containing a textual description of the grounds for denying the request

DESCRIPTION

This service informs a federate that a restoration request that was the subject of a previous `requestFederationRestore()` invocation by the federate has been denied. Restoration requests are arbitrated by the FedExec and may be denied for the following reasons:

- a race condition occurred among multiple federates attempting to initiate a save or restore and the federate's request lost
- a valid set of saved LRC states was not located in the directory given by the RID file parameter FullPathOfSaveDirectory under the RTI_FederationExecutive.
- the checksum of the FED file associated with the saved federation state is not the same as the checksum of the FED file used by the currently active federation
- there does not exist a function for mapping federate handles in the saved state to federate handles in the active federation at the time of the restoration such that:
  - the function is total, one-to-one, and onto
  - all mappings in the function are between federates of the same type

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges that the restoration request has failed.

EXCEPTIONS

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador
requestFederationRestoreSucceeded()
RTI::RTIambassador
requestFederationRestore()

B.1.11 requestFederationRestoreSucceeded()

RTI 1.3-NG

ABSTRACT

This service informs a federate that a request to attempt to restore its federation state has been accepted

HLA IF SPECIFICATION

This method (in conjunction with `requestFederationRestoreFailed()`) realizes the "Confirm Federation Restoration Request" Federation Management service as specified in the *HLA Interface Specification* (§4.17 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   requestFederationRestoreSucceeded (
      const char *label
   )
throw (
   RTI::FederateInternalError
)
```

ARGUMENTS

label

the save label associated with the successful restoration request

DESCRIPTION

This service informs a federate that a federation restoration request previously made by the federate has been accepted by the FedExec. The restoration process will proceed as detailed in the description of `requestFederationRestore()`.

Note that this callback does not imply that the restoration itself will ultimately succeed – only that the request meets the prerequisites.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges that the restoration request has been accepted.

EXCEPTIONS

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador

federationRestoreBegun()
requestFederationRestoreFailed()

RTI::RTIambassador

requestFederationRestore()

B.1.12 synchronizationPointRegistrationFailed()

RTI 1.3-NG

ABSTRACT

This service informs a federate that an attempt to register a synchronization point has failed.

HLA IF SPECIFICATION

This method (in conjunction with `synchronizationPointRegistrationSucceeded()`) realizes the "Confirm Synchronization Point Registration" Federation Management service as specified in the *HLA Interface Specification* (§4.7 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   synchronizationPointRegistrationFailed (
      const char *label
   )
throw (
   RTI::FederateInternalError
)
```

ARGUMENTS

label

the synchronization label associated with the request

DESCRIPTION

Synchronization points are a generalization of the pause/resume capabilities featured in early revisions of HLA. They provide a mechanism for federates to schedule checkpoints with federation-defined semantics, while relying on the RTI to perform the bookkeeping associated with determining when the checkpoint is achieved by the desired set of federates.

This callback informs a federate that an attempt by the federate to register a synchronization point using `registerFederationSynchronizationPoint()` did not succeed. An attempt to register a synchronization point may fail for the following reasons:

- the synchronization-point label is associated with an outstanding *universal* or *specified* synchronization point that has been announced to the federate through the `announceSynchronizationPoint()` callback
- the synchronization-point label is associated with an outstanding *specified* synchronization point that does not apply to the local federate
- a race condition occurred among multiple federates attempting to register synchronization points with the same label but different federate-sets, and the local federate's request lost

If the registration failed because of a race condition, the `synchronizationPointRegistrationFailed()` callback will occur asynchronously with respect to the `registerFederationSynchronizationPoint()` request (i.e., during a subsequent invocation of `tick()`). Otherwise, the callback will occur synchronously with respect to the service invocation; that is to say, before the return of `registerFederationSynchronizationPoint()` has returned.

Upon receipt of such a callback, the federate may wish to reregister the synchronization point with a different label or wait until the existing synchronization point has been achieved.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges that the synchronization-point registration has failed.

EXCEPTIONS

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador

synchronizationPointRegistrationSucceeded()

RTI::RTIambassador

registerFederationSynchronizationPoint()

B.1.13 synchronizationPointRegistrationSucceeded()

RTI 1.3-NG

ABSTRACT

This service informs a federate that an attempt to register a synchronization point has succeeded.

HLA IF SPECIFICATION

This method (in conjunction with `synchronizationPointRegistrationFailed()`) realizes the "Confirm Synchronization Point Registration" Federation Management service as specified in the *HLA Interface Specification* (§4.7 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   synchronizationPointRegistrationSucceeded (
      const char *label
   )
throw (
   RTI::FederateInternalError
)
```

ARGUMENTS

label

the synchronization label associated with the request

DESCRIPTION

Synchronization points are a generalization of the pause/resume capabilities featured in early revisions of HLA. They provide a mechanism for federates to schedule checkpoints with federation-defined semantics, while relying on the RTI to perform the bookkeeping associated with determining when the checkpoint is achieved by the desired set of federates.

This callback informs a federate that an attempt by the federate to register a synchronization point using `registerFederationSynchronizationPoint()` has succeeded. The synchronization process will proceed as detailed in the description of `registerFederationSynchronizationPoint()`.

Note that such a callback does not imply that the synchronization has occurred – only that the synchronization point has been accepted.

This callback always occurs during a subsequent invocation of `tick()` and never synchronously with respect to the `registerFederationSynchronizationPoint()` service invocation.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges that the synchronization-point registration has succeeded.

EXCEPTIONS

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador

announceSynchronizationPoint()
synchronizationPointRegistrationFailed()

RTI::RTIambassador

registerFederationSynchronizationPoint()

B.2 Declaration Management

B.2.1 startRegistrationForObjectClass()

RTI 1.3-NG

ABSTRACT

This service advises a federate of the existence of an active subscriber for an object class that is published by the federate.

HLA IF SPECIFICATION

This method realizes the "Start Registration For Object Class" Declaration Management service as specified in the *HLA Interface Specification* (§5.10 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   startRegistrationForObjectClass (
      RTI::ObjectClassHandle   theClass
   )
throw (
   RTI::ObjectClassNotPublished,
   RTI::FederateInternalError
)
```

ARGUMENTS

theClass

the published object class for which remote subscribers exist

DESCRIPTION

The `startRegistrationForObjectClass()` callback advises the federate that a remote federate is actively subscribed to the specified object class (or a superclass) such that the set of subscribed class-attributes intersects the set of class-attributes published by the local federate. Subscription regions are not considered when making such an advisory. Upon receipt of such a callback, the federate should begin registration and update of instances of the specified object class.

A `stopRegistrationForObjectClass()` callback will be made if all remote active subscribers have unsubscribed or resigned. At this point, the federate may safely cease registering and updating instances of the class. Any such activity would be superfluous in the absence of subscribers. These callbacks are made strictly for advisory purposes and are not enforced in any way by the RTI.

A federate may toggle the generation of registration advisories by its local LRC on and off for the `enableClassRelevanceAdvisorySwitch()` and `disableClassRelevanceAdvisorySwitch()` services, respectively.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the advisory.

EXCEPTIONS

RTI::ObjectClassNotPublished

The operation attempted requires that the object class be currently published by the federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::

stopRegistrationForObjectClass()

RTI::RTIambassador::

disableClassRelevanceAdvisorySwitch()
enableClassRelevanceAdvisorySwitch()
getObjectClassName()
subscribeObjectClassAttributes()

B.2.2 stopRegistrationForObjectClass()

RTI 1.3-NG

ABSTRACT

This service advises a federate of the absence of active subscribers for an object class that is published by the federate.

HLA IF SPECIFICATION

This method realizes the "Stop Registration For Object Class" Declaration Management service as specified in the *HLA Interface Specification* (§5.11 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   stopRegistrationForObjectClass (
      RTI::ObjectClassHandle   theClass)
throw (
   RTI::ObjectClassNotPublished,
   RTI::FederateInternalError
 )
```

ARGUMENTS

theClass

the object class for which there are no active subscribers

DESCRIPTION

The `startRegistrationForObjectClass()` callback advises the federate that no remote federates are actively subscribed to the specified object class (or a superclass) such that the set of subscribed class-attributes intersects the set of class-attributes published by the local federate. Subscription regions are not considered when making such an advisory. Upon receipt of such a callback, the federate may safely cease registering and updating instances of the class. Any such activity would be superfluous in the absence of subscribers.

Note that this callback is only made for object classes that have previously been the subject of a `startRegistrationForObjectClass()` callback. A newly publishing federate should assume the absence of subscribers until it receives such a callback.

A `startRegistrationForObjectClass()` callback will be made if a remote federate makes a subscription that intersects the local federate's publication. At this point, the federate should begin registration and update of instances of the specified object class. These callbacks are made strictly for advisory purposes and are not enforced in any way by the RTI.

A federate may toggle the generation of registration advisories by its local LRC on and off for using the `enableClassRelevanceAdvisorySwitch()` and `disableClassRelevanceAdvisorySwitch()` services, respectively.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the advisory.

EXCEPTIONS

RTI::ObjectClassNotPublished

The operation attempted requires that the object class be currently published by the federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
startRegistrationForObjectClass()
RTI::RTIambassador::
disableClassRelevanceAdvisorySwitch()
enableClassRelevanceAdvisorySwitch()
getObjectClassName()
subscribeObjectClassAttributes()

B.2.3 turnInteractionsOff()

RTI 1.3-NG

ABSTRACT

This service advises a federate of the absence of active subscribers for an interaction class published by the federate.

HLA IF SPECIFICATION

This service realizes the "Turn Interactions Off" Declaration Management service as specified in the *HLA Interface Specification* (§5.13 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   turnInteractionsOff (
      RTI::InteractionClassHandle theHandle
   )
throw (
   RTI::InteractionClassNotPublished,
   RTI::FederateInternalError
)
```

ARGUMENTS

theHandle

the interaction class for which there are no remote active subscribers

DESCRIPTION

The `turnInteractionsOff()` callback advises the federate that no remote federates are currently actively subscribing to the specified interaction class or a subclass. The federate may safely refrain from generating interactions of the specified class: such interactions would be superfluous in the absence of an active subscriber. Subscription regions are not considered when making such an advisory.

Note that this callback is only made for interaction classes that have previously been the subject of a `turnInteractionsOn()` callback. A newly publishing federate should assume the absence of subscribers until it receives such a callback.

A `turnInteractionsOn()` callback will be made if a remote federate subsequently actively subscribes to the interaction class or a subclass. At this point, the federate should begin generation of instances of the specified interaction class. These callbacks are made strictly for advisory purposes and are not enforced in any way by the RTI.

A federate may toggle the generation of interaction class advisories by its local LRC on and off using the `enableInteractionRelevanceAdvisorySwitch()` and `disableInteractionRelevanceAdvisorySwitch()` services, respectively.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the advisory.

EXCEPTIONS

RTI::InteractionClassNotPublished

The operation attempted requires that the interaction class be currently published by the federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
turnInteractionsOn()
RTI::RTIambassador::
disableInteractionRelevanceAdvisorySwitch()
enableInteractionRelevanceAdvisorySwitch()
getInteractionClassName()
publishInteractionClass()
subscribeInteractionClass()

B.2.4 turnInteractionsOn()

RTI 1.3-NG

ABSTRACT

This service advises a federate of the presence of active subscribers for an interaction class published by the federate.

HLA IF SPECIFICATION

This method realizes the "Turn Interactions On" Declaration Management service as specified in the *HLA Interface Specification* (§5.12 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   turnInteractionsOn (
      RTI::InteractionClassHandle theHandle
   )
throw (
   RTI::InteractionClassNotPublished,
   RTI::FederateInternalError
)
```

ARGUMENTS

theHandle

the interaction class for which active subscribers exist

DESCRIPTION

The `turnInteractionsOn()` callback advises the federate that a remote federate is actively subscribed to the specified interaction class or a superclass. Subscription regions are not considered when making such an advisory. Upon receipt of such a callback, the federate should begin generating instances of the specified interaction class.

A `turnInteractionsOff()` callback will be made if all remote active subscribers have unsubscribed or resigned. At this point, the federate may safely cease generating instances of the interaction class: any such activity would be superfluous in the absence of subscribers. These callbacks are made strictly for advisory purposes and are not enforced in any way by the RTI.

A federate may toggle the generation of interaction class advisories by its local LRC on and off using the `enableInteractionRelevanceAdvisorySwitch()` and `disableInteractionRelevanceAdvisorySwitch()` services, respectively.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the advisory.

EXCEPTIONS

RTI::InteractionClassNotPublished

The operation attempted requires that the interaction class be currently published by the federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::

turnInteractionsOff()

RTI::RTIambassador::

disableInteractionRelevanceAdvisorySwitch()
enableInteractionRelevanceAdvisorySwitch()
getInteractionClassName()
publishInteractionClass()
subscribeInteractionClass()

B.3 Object Management

B.3.1 attributesInScope()

RTI 1.3-NG

ABSTRACT

This callback advises a federate that a set of instance-attributes of a specified object instance is associated with update regions intersecting the relevant subscription regions of the federate. The LRC may subsequently deliver reflections of the in-scope instance-attributes to the federate.

HLA IF SPECIFICATION

This method realizes the "Attributes In Scope" Federation Management service as specified in the *HLA Interface Specification* (§6.13 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   attributesInScope (
       RTI::ObjectHandle     theObject,
       const RTI::AttributeHandleSet& theAttributes
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object instance whose instance attributes are in-scope for the federate

theAttributes

the instance-attributes which are in-scope for the federate

DESCRIPTION

This callback informs the federate that reflections of the specified instance-attributes are relevant to the federate based on the update regions associated with the instance-attributes and the federate's current subscription interests. The LRC may subsequently deliver `reflectAttributeValues()` callbacks for the in-scope instance-attributes. An instance-attribute is considered in-scope for a federate if and only if:

- the federate has discovered the object instance
- the federate is subscribing to the class-attribute corresponding to the instance-attribute *at the level of the discovered class of the object instance*
- the subscription region associated with the class-attribute intersects the update region associated with the instance-attribute (if the class-attribute is subscribed with the default region or the instance-attribute is associated with the default region for update, this is trivially the case)
- the federate does not own the instance-attribute

If an instance-attribute subsequently goes out of scope due to a change in subscription or update regions, the federate will receive an `attributesOutOfScope()` advisory.

The federate may toggle the generation of attribute-scope advisories on and off using the `enableAttributeScopeAdvisorySwitch()` and `disableAttributeScopeAdvisorySwitch()` services, respectively. The federate *will not* receive advisories retroactively for instance-attributes that go into or out-of scope while advisories are disabled.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the attribute scope advisory.

EXCEPTIONS

RTI::ObjectNotKnown

The specified object ID is not valid within the current FedExecec or is not known to the federate.

RTI::AttributeNotKnown

One or more of the specified attribute handles is not valid within the context of the specified object class.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
attributesOutOfScope()
discoverObjectInstance()
reflectAttributeValues()
RTI::RTIambassador::
associateRegionForUpdates()
disableAttributeScopeAdvisorySwitch()
enableAttributeScopeAdvisorySwitch()
registerObjectInstance()
registerObjectInstanceWithRegion()
subscribeObjectClassAttributes()
subscribeObjectClassAttributesWithRegion()

B.3.2 attributesOutOfScope()

RTI 1.3-NG

ABSTRACT

This callback advises a federate that a set of instance-attributes of a specified object instance is no longer in-scope for the federate due to changes in subscription or in the update regions associated with the instance-attributes. The LRC will no longer deliver reflections of the out-of-scope instance-attributes to the federate.

HLA IF SPECIFICATION

This method realizes the "Attributes Out Of Scope" Federation Management service as specified in the *HLA Interface Specification* (§6.14 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual void attributesOutOfScope (
      RTI::ObjectHandle        theObject,
const RTI::AttributeHandleSet& theAttributes
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object instance whose instance attributes are out-of-scope for the federate

theAttributes

the instance-attributes which are out-of-scope for the federate

DESCRIPTION

This callback informs the federate that the specified instance-attributes (which have previously been the subject of an `attributesInScope()` advisory) are no longer in scope. This can occur if the relevant subscription or update regions are changed such that they no longer intersect. The LRC will not deliver any reflections of the out-of-scope instance-attributes without an intervening `attributesInScope()` advisory.

The federate may toggle the generation of attribute-scope advisories on and off using the `enableAttributeScopeAdvisorySwitch()` and `disableAttributeScopeAdvisorySwitch()` services, respectively. The federate *will not* receive advisories retroactively for instance-attributes that go into or out-of scope while advisories are disabled.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the attribute-scope advisory.

EXCEPTIONS

RTI::ObjectNotKnown

The specified object ID is not valid within the current FedExecec or is not known to the federate.

RTI::AttributeNotKnown

One or more of the specified attribute handles is not valid within the context of the specified object class.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
attributesInScope()
RTI::RTIambassador::
disableAttributeScopeAdvisorySwitch()
enableAttributeScopeAdvisorySwitch()

B.3.3 discoverObjectInstance()

RTI 1.3-NG

ABSTRACT

This callback informs a federate of the existence of an object instance in the federation that is relevant to the federate's current subscription interests.

HLA IF SPECIFICATION

This service realizes the "Discover Object Instance" Federation Management service as specified in the *HLA Interface Specification* (§6.3 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   discoverObjectInstance (
      RTI::ObjectHandle   theObject,
      const char* RTI::ObjectClassHandle theObjectClass
      const char*  theObjectName
   )
throw (
   RTI::CouldNotDiscover,
   RTI::ObjectClassNotKnown,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

a numeric handle which may be used by the discovering federate to uniquely refer to the object instance

theObjectClass

the object class level at which the object instance is discovered

theObjectName

a symbolic name associated with the discovered object

DESCRIPTION

This callback informs a federate of the existence of an object instance in the federation that is relevant to the federate's current subscription interest. This discovery may occur as a result of any of the following:

- a `registerObjectInstance()` or `registerObjectInstanceWithRegion()` service invocation by a remote federate
- a modification of the associated update regions for instance-attributes of an object instance which causes the object instance to become relevant to the local federate
- a state update for instance-attributes of an object instance relevant to the local federate, but for which the local federate has not already received a discovery (e.g., if the local federate was not subscribed at the time of object registration)

The object class by which the object instance is discovered will be the registered object class of the instance or the most-specific superclass of the registered object class such that

- the discovering federate is currently subscribing to the object class
- the discovering federate is subscribing to at least one class-attribute at the level of the object class such that the corresponding instance-attribute of the object instance is currently owned in the federation and the region of subscription intersects the update region associated with the corresponding instance-attribute

Only a federate's subscriptions at the level of the discovered object class are considered in assessing the relevance of future updates instance-attributes of the discovered object instance. Ramifications of this include

- if subscriptions or update regions related to the object instance subsequently change such that the object instance's region of relevance intersects the federate's subscription interests at a more-specific object class, the instance will *not* be rediscovered as the more-specific object class, nor will updates of instance-attributes exclusive to the more-specific object class be presented to the federate
- an instance-attribute is not considered relevant if a federate's subscription to the class-attribute *at the level of the discovered class* does not intersect the instance-attribute's update region, even if the federate is subscribing to the class-attribute at some other level with a subscription region intersecting the update region

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the existence of the object instance in the federation.

An exceptional will result in an entry being made to the federate's log file; the object instance is still considered to have been discovered by the federate.

EXCEPTIONS

RTI::CouldNotDiscover

The federate was unable to discover the object for reasons other than an invalid object class or an error internal to the federate.

RTI::ObjectClassNotKnown

The object class handle is not valid in the context of the federation or is not currently subscribed by the federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
attributesInScope()
reflectAttributeValues()
removeObjectInstance()

RTI::RTIambassador::
getObjectClassName()
getObjectInstanceName()
localDeleteObjectInstance()
registerObjectInstance()
registerObjectInstanceWithRegion()
requestObjectAttributeValueUpdate()
subscribeObjectClassAttributes()
subscribeObjectClassAttributesWithRegion()

B.3.4 provideAttributeValueUpdate()

RTI 1.3-NG

ABSTRACT

This callback informs the federate that an update of a set of locally owned instance-attributes of a specified object instance has been solicited.

HLA IF SPECIFICATION

This method realizes the “Provide Attribute Value Update” Object Management service as specified in the *HLA Interface Specification* (§6.16 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   provideAttributeValueUpdate (
         RTI::ObjectHandle theObject,
  const RTI::AttributeHandleSet& theAttributes
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::AttributeNotOwned,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

Object ID of the object for whom an attribute-value update is requested.

theAttributes

Set of attributes for which an update is requested; these must be attributes whose ownership tokens are held by the federate. The caller maintains ownership of the storage associated with this set; the federate ambassador should make a copy if it wishes to retain this information after the completion of the call.

DESCRIPTION

This callback is invoked to notify the federate that an attribute-update has been requested by another federate via its *RTIambassador::requestClassAttributeValueUpdate* or *RTIambassador::requestObjectAttributeValueUpdate* service. (Note that one attribute-value request can trigger multiple provide attribute-value update callbacks on different federates.)

Upon receipt of such a request, the federate should update the specified object-attributes (using *RTIambassador::updateAttributeValues*) as soon as possible. (Keep in mind that this may not be done from inside the *FederateAmbassador::provideAttributeValueUpdate* callback as this would result in a concurrent access violation.)

RETURN VALUES

A non-exceptional return indicates that the federate understands the attribute value update request and intends to update the specified object-attributes.

An exceptional return will cause an entry to be made in the federate's RTI log file; the federate is still responsible for updating the requested attributes.

EXCEPTIONS

RTI::ObjectNotKnown

The specified object ID is not valid within the current FedExecec or is not known to the federate.

RTI::AttributeNotKnown

One or more of the attribute handles is not valid in the context of the specified object or the federate does not hold the attribute's ownership token.

RTI::AttributeNotOwned

One or more of the specified attribute-instances is not owned by the local federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::RTIambassador::

requestObjectAttributeValueUpdate()
updateAttributeValues()

B.3.5 receiveInteraction()

RTI 1.3-NG

ABSTRACT

This callback informs a federate of an interaction in the federation that is relevant to the federate's current subscription interests.

HLA IF SPECIFICATION

This method realizes the "Receive Interaction" Object Management service as specified in the *HLA Interface Specification* (§6.7 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   receiveInteraction (
       RTI::InteractionClassHandle   theInteraction,
       const RTI::ParameterHandleValuePairSet&
           theParameters,
       const RTI::FedTime&           theTime,
       const char                    *theTag,
       RTI::EventRetractionHandle    theHandle
   )
throw (
   RTI::InteractionClassNotKnown,
   RTI::InteractionParameterNotKnown,
   RTI::InvalidFederationTime,
   RTI::FederateInternalError
)

virtual
void
RTI::FederateAmbassador::
   receiveInteraction (
       RTI::InteractionClassHandle  theInteraction,
       const RTI::ParameterHandleValuePairSet&
           theParameters,
       const char                   *theTag
   )
throw (
   RTI::InteractionClassNotKnown,
   RTI::InteractionParameterNotKnown
   RTI::FederateInternalError
)
```

ARGUMENTS

theInteraction

the interaction class of the interaction instance

theParameters

the set of parameters associated with the interaction instance

theTime

the logical time used for time-stamp-ordered (TSO) delivery of the interaction

theTag

the user-specified tag passed to the invocation of `sendInteraction()` resulting in the receipt

theHandle

the event handle that uniquely identifies the TSO event for purposes of retraction

DESCRIPTION

The callback is invoked to notify the federate of an interaction in the federation relevant to the current subscription interests of the federate. Interactions may be produced by the RTI itself, or they may be produced by federates using the `sendInteraction()` or `sendInteractionWithRegion()` service.

An interaction will be delivered to a federate as the actual class of the interaction or the most-specific superclass of the actual class of the interaction such that the federate's subscription region intersects the region associated with the interaction instance.

An interaction instance will be delivered as time-stamp-ordered (TSO) if any only if

- The federate initiating the interaction is time regulating at the time the interaction is sent.
- A logical time argument is provided to the `sendInteraction()` service invocation resulting in the interaction.
- At the initiating federate, the interaction class is associated with a TSO ordering service in the FED file or through a subsequent `changeInteractionOrderType()` service invocation.
- The receiving federate is time-constrained at the point at which the interaction is received and the point at which the interaction is delivered.

A logical time argument will be provided to `receiveInteraction()` if only if the interaction instance is delivered in time-stamp-order (even if a logical time was provided to the `sendInteraction()` invocation initiating the interaction.)

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the receipt of the interaction.

An exceptional return will cause an entry to be made in the federate's RTI log file; the interaction is still considered to have been delivered to the federate.

EXCEPTIONS

RTI::InteractionClassNotKnown

The specified interaction class handle is not valid in the context of the current Federation Execution or is not subscribed by the federate.

RTI::InteractionParameterNotKnown

One or more of the specified parameters handles is not valid in the context of the specified interaction class.

RTI::InvalidFederationTime

The federation time is not valid, i.e. a time-stamped-ordered interaction has been delivered to a time-constrained federation in the federate's past. (In 1.0 there's no way for the federate to tell whether the receipt is the result of a time-stamp-ordered interaction, so it's unclear when this exception should be raised.)

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
requestRetraction()
reflectAttributeValues()
RTI::ParameterHandleValuePairSet
RTI::RTIambassador::
sendInteraction()
sendInteractionWithRegion()
subscribeInteractionClass()
subscribeInteractionClassWithRegion()
tick()

B.3.6 reflectAttributeValues()

RTI 1.3-NG

ABSTRACT

This callback informs the federate of a state update for a set of instance-attributes relevant to the federate's current subscription interests.

HLA IF SPECIFICATION

This method realizes the "Reflect Attribute Values" Object Management service as specified in the *HLA Interface Specification* (§6.5 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   reflectAttributeValues (
       RTI::ObjectHandle    theObject,
       const RTI::AttributeHandleValuePairSet&
           theAttributes,
       const RTI::FedTime&     theTime,
       const char              *theTag,
       RTI::EventRetractionHandle        theHandle
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::FederateOwnsAttributes,
   RTI::InvalidFederationTime,
   RTI::FederateInternalError
)

virtual
void
reflectAttributeValues (
       RTI::ObjectHandle                 theObject,
       const RTI::AttributeHandleValuePairSet&
           theAttributes,
       const char                        *theTag
)
throw (
   RTI::ObjectNotKnown
   RTI::AttributeNotKnown,
   RTI::FederateOwnsAttributes,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

object instance for whose is being updated

theAttributes

the instance-attributes whose state is being updated

theTime

the time stamp used for time-stamp-ordered delivery of the reflection

theTag

the user-specified tag that was passed to the invocation of `updateAttributeValues()` resulting in the reflection

theHandle

the handle uniquely identifying the time-stamp-ordered event for purposes of retraction

DESCRIPTION

This callback is invoked to communicate a change in state of some set of instance-attributes relevant to the federate's subscription interest. Attributes are used to represent persistent characteristics of federation state (e.g. the position of an entity.) Such a reflection is usually the result of an invocation of the `updateAttributeValues()` service by a remote federate. A single invocation of `updateAttributeValues()` may result in multiple reflections: one for each region/transport/order combination for the instance-attribute subjects of the update.

The object instance subject of a reflection must have previously been the subject of a `discoverObjectInstance()` callback. The instance-attribute subjects of a reflection must have previously been the subject of an `attributesInScope()` callback.

An instance-attribute that has been updated will only be reflected to a given federate if

- the federate is subscribing to the corresponding class-attribute *at the level of the object class by which the object-instance has been discovered by the federate*
- the subscription region of the aforementioned class-attribute intersects the update region associated with the instance-attribute
- the update was initiated by a different federate or by the RTI itself

A reflection will be delivered as time-stamp-ordered (TSO) to a federate if any only if:

- The federate initiating the update is time regulating at the time the update is sent.
- A logical time argument is provided to the `updateAttributeValues()` service invocation resulting in the interaction.
- At the initiating federate, the instance-attribute is associated with a TSO ordering service in the FED file or through a subsequent `changeAttributeOrderType()` service invocation.
- The receiving federate is time-constrained at the point at which the update is received and the point at which the update is delivered.

A logical time argument will be provided to `reflectAttributeValues()` only if the reflection is delivered in time-stamp-order (even if a logical time was provided to the `updateAttributeValues()` invocation resulting in the reflection.)

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the update of the instance-attributes.

An exceptional return will cause an error message to be written to the federate's RTI log file; the attribute values are still considered to have been reflected.

EXCEPTIONS

RTI::ObjectNotKnown

The object ID is has not previously been discovered by the federate.

RTI::AttributeNotKnown

One or more attribute handles are not valid in the context of the current Federation Execution or the attributes are not subscribed by the federate.

RTI::FederateOwnsAttributes

One or more attributes of the specified object are owned by the local federate.

RTI::InvalidFederationTime

The federation time is not valid, i.e. a time-stamped-ordered update has been delivered to a time-constrained federation in the federate's past. (In 1.0 there's no way for the federate to

tell whether the reflection is the result of a time-stamp-ordered update, so it's unclear when this exception should be raised.)

RTI::FederateInternalError
An error internal to the federate has occurred.

SEE ALSO

RTI::AttributeHandleValuePairSet
RTI::FederateAmbassador::
discoverObject()
reflectRetraction()
receiveInteraction()
RTI::RTIambassador::
updateAttributeValues()
enableTimeConstrained()
subscribeObjectClass()
timeAdvanceRequest()

B.3.7 removeObjectInstance()

RTI 1.3-NG

ABSTRACT

This callback informs a federate that an object instance no longer exists in the federation.

HLA IF SPECIFICATION

This method realizes the "Remove Object Instance" Federation Management service as specified in the *HLA Interface Specification* (§6.9 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   removeObjectInstance (
       RTI::ObjectHandle          theObject,
       const RTI::FedTime&        theTime,
       const char                 *theTag,
       RTI::EventRetractionHandle theHandle
   )
throw (
   RTI::ObjectNotKnown,
   RTI::InvalidFederationTime,
   RTI::FederateInternalError
)

virtual
void
RTI::FederateAmbassador::
   removeObjectInstance (
       RTI::ObjectHandle          theObject,
       const char                 *theTag
   )
throw (
   RTI::ObjectNotKnown,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object instance being removed

theTime

the time-stamp associated with a time-stamp-ordered object removal

theTag

the user-specified tag provided to the invocation of `deleteObjectInstance()` initiating the removal, or "Deleted by Resignation" if the instance was implicitly deleted due to resignation

theHandle

the event handle associated with a time-stamp-ordered deletion

DESCRIPTION

This callback informs the federate that the specified object instance has been deleted from the federation. This deletion may be the result of a `deleteObjectInstance()` service invocation by a remote federation, or of a resignation by a remote federate with a "delete objects" policy.

A `removeObjectInstance()` callback resulting from a `deleteObjectInstance()` service invocation will be delivered time-stamp-ordered (TSO) if and only if

- a logical time argument is provided to the `deleteObjectInstance()` service invocation
- a TSO delivery policy is in effect for the *privilegeToDelete* attribute of the specified object instance at the initiating federate
- the federate initiating the deletion is time regulating
- the federate receiving the deletion is time constrained at the point in execution at which the deletion is queued for delivery *and* the point in execution at which the deletion is delivered to the federate

No logical time argument will be provided to `removeObjectInstance()` callbacks that are not delivered in TSO order, even if a logical time was specified to the `deleteObjectInstance()` service invocation which initiated the deletion.

The removed object instance may be subsequently rediscovered if there are updates for the object-instance queued or in-transit at the time of the deletion.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the deletion of the specified object instance.

An exceptional return will cause an entry to be made in the federate's log file; the object instance is still considered deleted with respect to the federate.

EXCEPTIONS

RTI::ObjectNotKnown

The specified object ID is not valid within the current FedExecec or is not known to the federate.

RTI::InvalidFederationTime

The specified logical time argument does not represent a valid point on the federation time axis.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
discoverObjectInstance()
RTI::RTIambassador::
deleteObjectInstance()
enableTimeConstrained()
localDeleteObjectInstance()
resignFederationExecution()
tick()

B.3.8 turnUpdatesOffForObjectInstance()

RTI 1.3-NG

ABSTRACT

This service advises a federate that there are no active subscribers for a set of instance-attributes of a specified object instance.

HLA IF SPECIFICATION

This method realizes the "Turn Updates Off For Object Instance" Federation Management service as specified in the *HLA Interface Specification* (§6.18 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   turnUpdatesOffForObjectInstance (
         RTI::ObjectHandle        theObject,
  const RTI::AttributeHandleSet& theAttributes
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotOwned,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object instance to which the specified instance-attributes belong

theAttributes

the set of instance-attributes for which the federate is advised to cease updates

DESCRIPTION

This callback advises the federate that the specified instance-attributes (which have been the previous subject of `turnUpdatesOnForObjectInstance()` callbacks) are no longer actively subscribed by a remote federate. This advisory may be the result of a change in subscription by a remote federate, or a change in the update regions associated with the instance-attributes at the local federate.

The federate may safely cease updating the specified instance-attributes, as any subsequent updates would be superfluous in the absence of an active subscriber. The federate will be subsequently notified using `turnUpdatesOnForObjectInstance()` if an active subscriber reemerges for the instance-attributes.

The federate may toggle attribute-update advisories on and off using the `enableAttributeRelevanceAdvisorySwitch()` or `disableAttributeRelevanceAdvisorySwitch()` service, respectively. The federate will *not* be retroactively notified of changes in instance-attribute update relevance that occur while attribute-update advising is disabled.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the attribute-update advisory.

EXCEPTIONS

RTI::ObjectNotKnown

The specified object ID is not valid within the current FedExecec or is not known to the federate.

RTI::AttributeNotOwned

One or more of the specified attribute-instances is not owned by the local federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
attributesOutOfScope()
turnUpdatesOnForObjectInstance()
RTI::RTIambassador::
associateRegionForUpdates()
unassociateRegionForUpdates()
unsubscribeObjectClass ()
unsubscribeObjectClassAttributes()

B.3.9 turnUpdatesOnForObjectInstance()

RTI 1.3-NG

ABSTRACT

This service advises a federate that there are active subscribers for a set of instance-attributes of a specified object instance.

HLA IF SPECIFICATION

This method realizes the "Turn Updates On For Object Instance" Federation Management service as specified in the *HLA Interface Specification* (§6.17 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   turnUpdatesOnForObjectInstance (
         RTI::ObjectHandle        theObject,
  const RTI::AttributeHandleSet& theAttributes
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotOwned,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object instance for which updates are advised

theAttributes

the instance-attributes for which updates are advised

DESCRIPTION

This service advises a federate that there are active subscribers for the specified set of instance-attributes. An instance-attribute is considered relevant to a remote federate if and only if:

- the federate is subscribing the corresponding class-attribute at the level of the discovered object class of the instance
- the region of subscription of the above class-attribute intersects the update region associated with the instance-attribute (if the region of subscription or update region is the default region, this is trivially the case)

If remote subscriptions change such that there are no longer any active subscribers for some or all instance-attributes that have been the subject of a `turnUpdatesOnForObjectInstance()` callback, the federate will receive a `turnUpdatesOffForObjectInstance()` callback advising it that it may cease updating the instance-attributes.

The federate may toggle attribute-update advisories on and off using the `enableAttributeRelevanceAdvisorySwitch()` or `disableAttributeRelevanceAdvisorySwitch()` service, respectively. The federate will *not* be retroactively notified of changes in instance-attribute update relevance that occur while attribute-update advising is disabled.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the attribute-update advisory.

EXCEPTIONS

RTI::ObjectNotKnown

The specified object ID is not valid within the current FedExecec or is not known to the federate.

RTI::AttributeNotOwned

One or more of the specified attribute-instances is not owned by the local federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
attributesInScope()
turnUpdatesOffForObjectInstance()
RTI::RTIambassador::
associateRegionForUpdates()
registerObjectInstance()
registerObjectInstanceWithRegion()
subscribeObjectClassAttributes()
subscribeObjectClassAttributesWithRegion()
unassociateRegionForUpdates()
updateAttributeValues()

B.4 Ownership Management

B.4.1 attributeIsNotOwned()

RTI 1.3-NG

ABSTRACT

This service informs a federate that an instance-attribute that was previously the subject of a `queryAttributeOwnership()` service invocation exists in the federation but is not currently owned by any federate.

HLA IF SPECIFICATION

This method (in conjunction with `informAttributeOwnership()` and `attributeOwnedByRTI()`) realizes the "Inform Attribute Ownership" Ownership Management service as specified in the *HLA Interface Specification* (§7.16 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   attributeIsNotOwned (
      RTI::ObjectHandle    theObject,
      RTI::AttributeHandle theAttribute
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object-instance whose instance-attribute is unowned

theAttribute

the instance-attribute that is unowned

DESCRIPTION

This service informs the federate that an instance-attribute is not currently owned by any federate, and that the ownership "token" associated with the instance-attribute is being managed by some LRC in the federation. This advisory is provided in response to a `queryAttributeOwnership()` service invocation by the federate. If the local LRC of the querying federate is managing the ownership token, this callback will occur synchronously with respect to the `queryAttributeOwnership()` invocation. Otherwise, this callback will occur during a subsequent invocation of `tick()` by the querying federate (i.e., the ownership query is sent to the federation and a response returned asynchronously).

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the ownership advisory.

EXCEPTIONS

RTI::ObjectNotKnown

The object ID specified does not correspond to an object that has been discovered by the federate.

RTI::AttributeNotKnown

One or more of the specified attribute handles is not valid within the context of the specified object class.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
attributeOwnedByRTI()
informAttributeOwnership()
RTI::RTIambassador::
getAttributeName()
queryAttributeOwnership()

B.4.2 attributeOwnedByRTI()

RTI 1.3-NG

ABSTRACT

This service informs a federate that an instance-attribute that was previously the subject of a `queryAttributeOwnership()` service invocation is currently owned internally by the RTI.

HLA IF SPECIFICATION

This method (in conjunction with `informAttributeOwnership()` and `attributeIsNotOwned()`) realizes the "Inform Attribute Ownership" Ownership Management service as specified in the *HLA Interface Specification* (§7.16 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   attributeOwnedByRTI (
       RTI::ObjectHandle    theObject,
       RTI::AttributeHandle theAttribute
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object-instance whose instance-attribute is unowned

theAttribute

the instance-attribute that is unowned

DESCRIPTION

This service informs the federate that an instance-attribute is currently owned and updated by the RTI itself. Typically, the instance-attribute in question is an attribute of a MOM or internal RTI object class. This advisory is provided in response to a `queryAttributeOwnership()` service invocation by the federate. If the local LRC of the querying federate is the owner of the instance-attribute (e.g., if it is an attribute of the *Manager.Federate* object class updated by the LRC on behalf of the federate), this callback will occur synchronously with respect to the `queryAttributeOwnership()` invocation. Otherwise, this callback will occur during a subsequent invocation of `tick()` by the querying federate (i.e., the ownership query is sent to the federation and a response returned asynchronously).

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the ownership advisory.

EXCEPTIONS

RTI::ObjectNotKnown

The object ID specified does not correspond to an object that has been discovered by the federate.

RTI::AttributeNotKnown

One or more of the specified attribute handles is not valid within the context of the specified object class.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
attributeIsNotOwned()
informAttributeOwnership()
RTI::RTIambassador::
queryAttributeOwnership()
getAttributeName()

B.4.3 attributeOwnershipAcquisitionNotification()

RTI 1.3-NG

ABSTRACT

This service informs a federate that the ownership of a set of instance-attributes of an object instance has been acquired. The federate should immediately begin updating the instance-attributes as appropriate.

HLA IF SPECIFICATION

This method realizes the "Attribute Ownership Acquisition Notification" Ownership Management service as specified in the *HLA Interface Specification* (§7.6 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   attributeOwnershipAcquisitionNotification (
         RTI::ObjectHandle        theObject,
  const RTI::AttributeHandleSet& securedAttributes)
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::AttributeAcquisitionWasNotRequested,
   RTI::AttributeAlreadyOwned,
   RTI::AttributeNotPublished,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object-instance whose instance-attributes have been acquired by the federate

securedAttributes

the set of instance-attributes that have been acquired by the federate

DESCRIPTION

This callback informs the federate that it has acquired ownership of a specified set of instance-attributes of a specified object instance. Each instance-attribute subject of such a callback should be

- not currently owned by the local federate
- the subject of an outstanding acquisition request by the local federate, either initiated by the local federate or in response to an ownership assumption request
- existent in the federation
- not owned by any federate, or the subject of an outstanding negotiated divestiture request by a remote federate, either initiated by the remote federate or in response to an ownership release request

The instance-attributes may be a subset of the instance-attributes for which the federate has outstanding acquisition requests for the specified object. The federate may receive multiple `attributeOwnershipAcquisitionNotification()` callbacks in response to a single acquisition request, as different remote federates may own different subsets of the specified instance-attributes.

This callback may occur synchronously with respect to the service invocation requesting the acquisition if one or more instance-attributes are unowned and tracked by the LRC of the acquiring federate. For any other instance-attributes, an acquisition request will be sent out to the federation and `attributeOwnershipAcquisitionNotification()` callbacks may be delivered asynchronously, during a subsequent invocation of the `tick()` service.

RETURN VALUES

A non-exceptional return indicates that the federate assumes ownership of the specified instance-attributes.

An exceptional return will still result in an entry being made to the federate's log file. The federate still assumes ownership of the instance-attributes.

NOTES

- Ownership acquisition may be the result of an `attributeOwnershipAcquisition()` or `attributeOwnershipAcquisitionIfAvailable()` service invocation by the local federate.

EXCEPTIONS

RTI::ObjectNotKnown

The federate has not discovered an object with the specified object ID.

RTI::AttributeNotKnown

One or more of the attribute handles are not valid within the context of the specified object, are already owned by the federate, or are not published or not subscribed by the federate.

RTI::AttributeAcquisitionWasNotRequested

One or more of the instances-attributes is not the subject of a currently outstanding `attributeOwnershipAcquisition()` request.

RTI::AttributeAlreadyOwned

One or more of the instance-attributes is already owned by the local federate.

RTI::AttributeNotPublished

One or more of the specified attributes are not currently published by the local federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::AttributeHandleSet
RTI::FederateAmbassador::
confirmAttributeOwnershipAcquisitionCancellation()
requestAttributeOwnershipAssumption()
requestAttributeOwnershipRelease()
RTI::RTIambassador::
attributeOwnershipAcquisition()
attributeOwnershipAcquisitionIfAvailable()
attributeOwnershipReleaseResponse()
cancelAttributeOwnershipAcquisition()

B.4.4 attributeOwnershipDivestitureNotification()

RTI 1.3-NG

ABSTRACT

This service informs a federate that it has been relieved of ownership responsibilities for a specified set of instance-attributes.

HLA IF SPECIFICATION

This method realizes the "Attribute Ownership Divestiture Notification" Ownership Management service as specified in the *HLA Interface Specification* (§7.5 and 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   attributeOwnershipDivestitureNotification (
         RTI::ObjectHandle theObject,
  const RTI::AttributeHandleSet& releasedAttributes
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::AttributeNotOwned,
   RTI::AttributeDivestitureWasNotRequested,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object-instance whose instance-attributes have been divested by the federate

releasedAttributes

the set of instance-attributes that have been divested by the federate

DESCRIPTION

This callback informs the federate that it has divested ownership of a specified set of instance-attributes of a specified object instance. Each instance-attribute subject of such a callback should be

- currently owned by the local federate
- the subject of an outstanding divestiture request by the local federate, either initiated by the local federate or in response to an ownership release request
- in the case of negotiated divestitures, the subject of an outstanding acquisition request by a remote federate, either initiated by the remote federate or in response to an ownership acquisition request

The instance-attributes may be a subset of the instance-attributes of the specified object for which the federate has outstanding divestiture requests. The federate may receive multiple `attributeOwnershipDivestitureNotification()` callbacks in response to a single divestiture request, as different remote federates may assume ownership of different subsets of the divested instance-attributes.

This callback may occur synchronously with respect to the service invocation requesting the divestiture if a negotiated divestiture is requested and there exists an outstanding ownership assumption request at the time of the divestiture. For any other instance-attributes, a divestiture request will be sent out to the federation and `attributeOwnershipDivestitureNotification()` callbacks may be delivered asynchronously, during a subsequent invocation of the `tick()` service.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the transfer of the specified instance-attributes to a remote federate.

An exceptional return will result in an entry being made into the federate's log file; the ownership of the specified instance-attributes will still be transferred.

RTI 1.3 NOTES

- For unconditional divestiture requests, no `attributeOwnershipDivestitureNotification()` will be made. A non-exceptional return from `unconditionalAttributeOwnershipDivestiture()` implies that the instance-attributes have been successfully divested.
- This callback will only be made because of `negotiatedAttributeOwnershipDivestiture()` service invocations; the `unconditionalAttributeOwnershipDivestiture()` and `attributeOwnershipReleaseResponse()` services may also be used to divest ownership, but do not result in callbacks.

EXCEPTIONS

RTI::ObjectNotKnown

The federate has not discovered an object with the specified object ID.

RTI::AttributeNotKnown

One or more of the attributes are not valid in the context of the specified object or have not been the subject of a divestiture request.

RTI::AttributeNotOwned

One or more of the specified attribute-instances is not owned by the local federate.

RTI::AttributeDivestitureWasNotRequested

One or more of the instance-attributes is not the subject of a currently outstanding `negotiatedAttributeOwnershipDivestiture()` request.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::AttributeHandleSet
RTI::FederateAmbassador::
requestAttributeOwnershipRelease()
RTI::RTIambassdaor::
attributeOwnershipReleaseResponse()
negotiatedAttributeOwnershipDivestiture()
requestAttributeOwnershipRelease()
unconditionalAttributeOwnershipDivestiture()

B.4.5 attributeOwnershipUnavailable()

RTI 1.3-NG

ABSTRACT

This service informs a federate that a set of instance-attributes of an object instance that the federate attempted to acquire are currently owned by remote federates or by the RTI itself.

HLA IF SPECIFICATION

This service realizes the "Attribute Ownership Unavailable" Ownership Management service as specified in the *HLA Interface Specification* (§7.9 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   attributeOwnershipUnavailable (
         RTI::ObjectHandle        theObject,
  const RTI::AttributeHandleSet& theAttributes
   )
throw (
   RTI::ObjectNotKnown
   RTI::AttributeNotKnown,
   RTI::AttributeNotDefined,
   RTI::AttributeAlreadyOwned,
   RTI::AttributeAcquisitionWasNotRequested,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object instance whose instance-attributes are currently owned

theAttributes

the instance-attributes that are currently owned

DESCRIPTION

This callback informs a federate that some or all instance-attributes of the specified object instance that are currently the subject of outstanding `attributeOwnershipAcquisitionIfAvailable()` requests are already owned by a remote federate or by the RTI itself. If an instance-attribute is owned by the RTI and updated by the acquiring federate's LRC (e.g., an attribute of the *Manager.Federate* object class updated on behalf of the federate), such a callback will be made synchronously with respect to the `attributeOwnershipAcquisitionIfAvailable()` service invocation. For all other instance-attributes, an acquisition request will be sent out to the federation and `attributeOwnershipUnavailable()` callbacks may be delivered asynchronously, during subsequent invocations of `tick()`. A single acquisition request may result in multiple `attributeOwnershipUnavailable()` callbacks, as different subsets of the requested attributes may be owned by different remote federates and LRCs.

Note that a federate will *not* receive `attributeOwnershipUnavailable()` callbacks for instance-attributes that are no longer in existence in the federation.

Upon receipt of such a callback, a federate may wish to use `attributeOwnershipAcquisition()` to solicit ownership of the attribute from the remote federate.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the ownership advisory.

EXCEPTIONS

RTI::ObjectNotKnown

The federate has not discovered an object with the specified object ID.

RTI::AttributeNotKnown

One or more of the attribute handles are not valid within the context of the specified object, are already owned by the federate, or are not published or not subscribed by the federate.

RTI::AttributeNotDefined

The attribute handle is not valid in the context of the current FedExecec.

RTI::AttributeAlreadyOwned

One or more of the instance-attributes is already owned by the local federate.

RTI::AttributeAcquisitionWasNotRequested

One or more of the instances-attributes is not the subject of a currently outstanding `attributeOwnershipAcquisition()` request.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::AttributeHandleSet
RTI::FederateAmbassador::
attributeOwnershipAcquisitionNotification()
RTI::RTIambassador::
attributeOwnershipAcquisitionIfAvailable()
getAttributeName()

B.4.6 confirmAttributeOwnershipAcquisitionCancellation()

RTI 1.3-NG

ABSTRACT

This service informs a federate that an outstanding request to cancel acquisition of a specified set of instance-attributes of a specified object instance has been achieved.

HLA IF SPECIFICATION

This method realizes the "Confirm Attribute Ownership Acquisition Cancellation" Ownership Management service as specified in the *HLA Interface Specification* (§7.14 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   confirmAttributeOwnershipAcquisitionCancellation (
         RTI::ObjectHandle        theObject,
  const RTI::AttributeHandleSet& theAttributes
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::AttributeNotDefined,
   RTI::AttributeAlreadyOwned,
   RTI::AttributeAcquisitionWasNotCanceled,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object-instance for which instance-attribute acquisition has been cancelled

theAttributes

the instance-attributes for which acquisition has been cancelled

DESCRIPTION

This callback informs the federate that a cancellation of ownership acquisition, as requested using the `cancelAttributeOwnershipAcquisition()` service, has succeeded. The federate will not receive `attributeOwnershipAcquisition()` callbacks for any of the specified instance-attributes because of the cancelled acquisition request.

An acquisition cancellation must be coordinated with the federation to guard against the race condition that would occur if an acquisition were cancelled while an ownership transfer response was in-transit. As such, `confirmAttributeOwnershipAcquisitionCancellation()` callbacks will always be delivered asynchronously to `cancelAttributeOwnershipAcquisition()` requests, during subsequent invocations of the `tick()` service.

Multiple `confirmAttributeOwnershipAcquisitionCancellation()` cancellations may be received for a single cancellation request, as different subsets of the cancelled attributes may be confirmed by different remote federates.

A federate will *not* receive a cancellation confirmation for

- instance-attributes that do not exist in the federation (no callback will be received)
- instance-attributes for which an ownership transfer was already in-transit from a remote federate when it received the cancellation request (an `attributeOwnershipAcquisition()` callback will be received instead)
- instance-attributes for which an ownership transfer was already in-transit from a remote federate *to a federate other than the canceling federate*, such that the transfer was initiated when the cancellation request was received by the sender and had not yet arrived when the cancellation request was received by the recipient (no callback will be received)

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the cancellation of instance-attribute ownership acquisition.

An exceptional return will result in an entry being made to the federate's log file; instance-attribute acquisition is still cancelled.

EXCEPTIONS

RTI::ObjectNotKnown

The object ID specified does not correspond to an object that has been discovered by the federate.

RTI::AttributeNotKnown

One or more of the specified attribute handles is not valid within the context of the specified object class.

RTI::AttributeNotDefined

The attribute handle is not valid in the context of the current FedExec.

RTI::AttributeAcquisitionWasNotCanceled

One or more of the attribute-instances is not the subject of a currently outstanding `cancelAttributeOwnershipAcquisition()` request.

RTI::AttributeAlreadyOwned

One or more of the instance-attributes is already owned by the local federate.

RTI::AttributeNotKnown

One or more of the specified attribute handles is not valid within the context of the specified object class.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::AttributeHandleSet
RTI::FederateAmbassador::
attributeOwnershipAcquisitionNotification()
RTI::RTIambassador::
attributeOwnershipAcquisition()
cancelNegotiatedAttributeOwnershipDivestiture()
getAttributeName()

B.4.7 informAttributeOwnership()

RTI 1.3-NG

ABSTRACT

This callback informs a federate as to which federate in the federation owns an instance-attribute. In addition, the RTI notifies a federate of attributes that are unowned or owned by the RTI itself, using *attributeIsNotOwned* or *attributeOwnedByRTI* callbacks, respectively.

HLA IF SPECIFICATION

The RTI 1.3 implementation of this method (in conjunction with `attributeIsNotOwned()` and `attributeOwnedByRTI()`) realizes the "Inform Attribute Ownership" Ownership Management service as specified in the *HLA Interface Specification* (§7.16 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   informAttributeOwnership (
      RTI::ObjectHandle    theObject,
      RTI::AttributeHandle theAttribute,
      RTI::FederateHandle  theOwner
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object instance whose instance-attribute is owned by the specified federate

theAttribute

the instance-attribute which is owned by the specified federate

theOwner

the federate which owns the specified instance-attribute

DESCRIPTION

This callback informs the federate that the specified instance-attribute is owned by the specified federate. This advisory is provided in response to a `queryAttributeOwnership()` service invocation by the federate.

If the instance-attribute is owned by the querying federate, this callback will occur synchronously with respect to the `queryAttributeOwnership()` invocation. Otherwise, this callback will occur during a subsequent invocation of `tick()` by the querying federate (i.e., the ownership query is sent to the federation and a response returned asynchronously).

Note that an instance-attribute for which a federate has outstanding negotiated divestiture requests is still considered to be held by the federate until ownership is assumed by another federate.

RETURN VALUES

A non-exceptional return from this service indicates that the federate acknowledges the ownership advisory.

RTI1.3-NG NOTES

- If an instance-attribute is unowned, the querying federate will receive an `attributeIsNotOwned()` callback.
- If an instance-attribute is owned by the RTI itself, the querying federate will receive an `attributeOwnedByRTI()` callback.
- If an instance-attribute is non-existent in the federation or is in-transit between federates when the query is made, the querying federate will receive no response callback.

EXCEPTIONS

RTI::ObjectNotKnown

The specified object handle does not correspond to an object known by the federate's object manager.

RTI::AttributeNotKnown

The specified attribute handle is not valid in the context of the specified object instance.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FederateAmbassador::
attributeIsNotOwned()
attributeOwnedByRTI()
RTI::RTIambassador::
queryAttributeOwnership()

B.4.8 requestAttributeOwnershipAssumption()

RTI 1.3-NG

ABSTRACT

This callback informs a federate of the fact that a specified set of instance-attributes of a specified object instance has become available for acquisition. The federate is requested to assume ownership of some or all of the instance-attributes, if possible.

HLA IF SPECIFICATION

This method realizes the "Request Attribute Ownership Assumption" Ownership Management service as specified in the *HLA Interface Specification* (§5.2 7.4 and 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   requestAttributeOwnershipAssumption (
         RTI::ObjectHandle theObject,
  const RTI::AttributeHandleSet& offeredAttributes,
  const char              *theTag
   )
throw (
   RTI::ObjectNotKnown
   RTI::AttributeNotKnown,
   RTI::AttributeAlreadyOwned,
   RTI::AttributeNotPublished,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object instance whose instance-attributes are available for acquisition

offeredAttributes

the instance-attributes which are available for acquisition

theTag

the string that was provided as an argument to the divestiture request (or the empty string if the divestiture occurred implicitly as a result of some other service invocation)

DESCRIPTION

This callback informs the federate that the specified instance-attributes have become unowned. A federate will only receive such a notification for an instance-attribute if the federate is publishing the corresponding class-attribute at the level of the object class by which the instance is discovered by the federate. The announcement may result from

- an explicit divestiture (negotiated or unconditional) by a remote federate
- a remote federate resignation using the *RTI::RELEASE_ATTRIBUTES* or *RTI::DELETE_OBJECTS_AND_RELEASE_ATTRIBUTES* ownership-resolution policy

The federate subsequently communicates to the RTI that instance-attributes (if any) it is willing to assume ownership of. If the federate is willing to assume ownership of any instance-attributes, this fact is communicated to the LRC of the divesting federate. Ownership will be transferred to the federate whose response is received first by the LRC of the releasing federate. If a federate receives ownership of any instance-attributes because of a response to an acquisition request, it will subsequently receive notification in the form of an `attributeOwnershipAcquisitionNotification()` callback.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the assumption request. See release notes for more details.

NOTES

- This callback may also occur as a result of an unpublication by a remote federate: any locally owned instance-attributes of object instances known to the federate as the unpublished class will be implicitly divested. Any such instance-attributes subsequently acquired by the unpublishing federate (because of ownership management services outstanding at the time of the unpublication) will also be implicitly divested as they are acquired.
- The federate communicates the set of instance-attributes it is willing to assume ownership of though a subsequent invocation of `attributeOwnershipAcquisition()` or `attributeOwnershipAcquisitionIfAvailable()`. No response is necessary if the federate is not willing to assume ownership of any instance-attributes.
- The federate will be notified of the instance-attributes it has acquired ownership of (if any) through the `attributeOwnershipAcquisitionNotification()` callback. If the federate responded using `attributeOwnershipAcquisitionIfAvailable()`, it may receive an `attributeOwnershipUnavailable()` callback for those instance-attributes that were not acquired. Otherwise, it will receive no subsequent callbacks related to instance-attributes that were not acquired.

EXCEPTIONS

RTI::ObjectNotKnown

The object ID does not correspond to an object previously discovered by the federate.

RTI::AttributeNotKnown

One or more of the specified attribute handles is not valid within the context of the specified object class.

RTI::AttributeAlreadyOwned

One or more of the attributes contained in the set is already owned by the federate.

RTI::AttributeNotPublished

One or more of the specified attributes are not currently published by the local federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::AttributeHandleSet
RTI::FederateAmbassador::
attributeOwnershipAcquisitionNotification()
attributeOwnershipUnavailable()
RTI::RTIambassador::
negotiatedAttributeOwnershipDivestiture()
publishObjectClass()
resignFederationExecution()
subscribeObjectClassAttributes()
unconditionalAttributeOwnershipDivestiture()
unpublishObjectClass()
unsubscribeObjectClass()

B.4.9 requestAttributeOwnershipRelease()

RTI 1.3-NG

ABSTRACT

This callback informs a federate of a request by a remote federate to acquire a specified set of instance-attributes of a specified object instance owned by the federate.

HLA IF SPECIFICATION

This method realizes the "Request Attribute Ownership Release" Ownership Management service as specified in the *HLA Interface Specification* (§7.10 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
requestAttributeOwnershipRelease (
        RTI::ObjectHandle    theObject,
  const RTI::AttributeHandleSet& candidateAttributes,
  const char                 *theTag
   )
throw (
   RTI::ObjectNotKnown,
   RTI::AttributeNotKnown,
   RTI::AttributeNotOwned,
   RTI::FederateInternalError
)
```

ARGUMENTS

theObject

the object instance whose instance-attributes have been requested

candidateAttributes

the instance-attributes which have been requested

theTag

the string that was provided as an argument to the acquisition request; this string is not interpreted by the RTI and may be used to encode federation-specified information about the acquisition request

DESCRIPTION

This callback informs the federate that the specified instance-attributes were the subject of an *intrusive* acquisition request made by another federate (i.e., a request to acquire instance-attributes even if they are already owned). Upon receipt of such, the federate should communicate to the RTI the subset of the instance-attributes for which it is willing to relinquish ownership. If this subset is non-empty, ownership of the instance-attributes will be immediately transferred to the requesting federate. The acquiring federate will be notified of such using the `attributeOwnershipAcquisitionNotification()` callback.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the ownership release request. See release notes for more details.

NOTES

- The RTI 1.3 `attributeOwnershipAcquisition()` service is used to make intrusive acquisition requests; the `attributeOwnershipAcquisitionIfAvailable()` service is used to make non-intrusive acquisition requests.
- The federate communicates the set of instance-attributes for which it is willing to release ownership using the `unconditionalAttributeOwnershipDivestiture()`, `negotiatedAttributeOwnershipDivestiture()`, or `attributeOwnershipReleaseResponse()` service. No response is necessary if the federate is unwilling to release any instance-attributes.
- If the federate responds using the `negotiatedAttributeOwnershipDivestiture()` service, it will immediately receive an `attributeOwnershipDivestitureNotification()` callback (i.e., before the `tick()` service invocation resulting in the `requestAttributeOwnershipRelease()` has returned.) Otherwise, the releasing federate will receive no further notification that the instance-attributes have been released.

EXCEPTIONS

RTI::ObjectNotKnown

The object ID specified does not correspond to an object that has been discovered by the federate.

RTI::AttributeNotKnown

One or more of the specified attribute handles is not valid in the context of the specified object or the attribute ownership token is not held by the federate.

RTI::AttributeNotOwned

One or more of the specified attribute-instances is not owned by the local federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::AttributeHandleSet
RTI::FederateAmbassador::
attributeOwnershipAcquisitionNotification()
attributeOwnershipDivestitureNotification()
RTI::RTIambassador::
attributeOwnershipAcquisition()
attributeOwnershipReleaseResponse()
negotiatedAttributeOwnershipDivestiture()
unconditionalAttributeOwnershipDivestiture()

B.5 Time Management

B.5.1 requestRetraction()

RTI 1.3-NG

ABSTRACT

This callback advises the federate that a previously delivered time-stamp-ordered (TSO) event has been retracted.

HLA IF SPECIFICATION

This method realizes the “Request Retraction” Time Management service as specified in the *HLA Interface Specification* (§8.22 in version 1.3). This service was an Object Management service in previous revisions of the specification.

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::RTIambassador::
   requestRetraction (
       RTI::EventRetractionHandle theHandle
)
throw (
  RTI::EventNotKnown,
  RTI::FederateInternalError
)
```

ARGUMENTS

theHandle

the event-retraction handle of the TSO event being retracted

DESCRIPTION

This callback advises the federate that a TSO event previously delivered to the federate has been retracted using the `retract()` service. If an event is still queued for delivery to a federate when a retraction arrives at the LRC, the event is removed from the queues and discarded without a `requestRetraction()` being made.

The RTI simply distributes event retraction notifications to the relevant federates; it is up to the federation to implement the desired event-retraction semantics.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the retraction of the specified TSO event.

EXCEPTIONS

RTI::EventNotKnown

The retraction handle does not correspond to an event previously delivered to the local federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::RTIambassador::

retract()

B.5.2 timeAdvanceGrant()

RTI 1.3-NG

ABSTRACT

This service informs a federate that a previous time advance request, flush queue request, or next event request has been completed.

HLA IF SPECIFICATION

This method realizes the "Time Advance Grant" Time Management service as specified in the *HLA Interface Specification* (§8.13 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   timeAdvanceGrant (
      const FedTime& theTime
   )
throw (
   RTI::InvalidFederationTime,
   RTI::TimeAdvanceWasNotInProgress,
   RTI::FederationTimeAlreadyPassed,
   RTI::FederateInternalError
)
```

ARGUMENTS

theTime

the logical time to which the federate has advanced as a result of the currently outstanding time-advancement service

DESCRIPTION

This callback advises the federate that the current time-advancement service (i.e., `timeAdvanceRequest()`, `timeAdvanceRequestAvailable()`, `nextEventRequest()`, `nextEventRequestAvailable()`, or `flushQueueRequest()`) has completed, as determined by the criteria established by the particular type of time-advancement service.

Subsequent to a `timeAdvanceGrant()` callback

- The federate may initiate another time-advancement service.
- The federate may enable time-regulation or time-constraint.
- The federate's logical time is equal to the specified grant time and will remain so until the invocation of another time-advancement service.
- No time-stamp-ordered (TSO) events with a time stamp less than the grant time will be subsequently delivered to the federate.
- If the `timeAdvanceRequest()` or `nextEventRequest()` service is in progress, no TSO events with a time stamp equal to the grant time will be subsequently delivered to the federate.

If the federate is time-constrained

- No TSO events will be delivered to the federate until a subsequent invocation of a time-advancement service.
- No receive-ordered (RO) events will be delivered to the federate until a subsequent invocation of a time-advancement service *unless* asynchronous delivery of RO events is enabled.

If a federate is non-time-constrained, the criteria for a time-advance grant will be trivially met.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges the completion of the outstanding time-advancement service.

An exceptional return will cause an entry to be made in the federate's RTI log file; the logical time of the federate is still advanced.

EXCEPTIONS

RTI::InvalidFederationTime

The specified grant time is invalid.

RTI::TimeAdvanceWasNotInProgress

There is not an outstanding time advance request, next event request, or flush queue request.

RTI::FederationTimeAlreadyPassed

The specified grant time is less than the current federate logical time.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FedTime
RTI::RTIambassador::
enableTimeConstrained()
enableTimeRegulation()
flushQueueRequest()
nextEventRequest()
queryFederateTime()
queryLBTS()
tick()
timeAdvanceRequest()

B.5.3 timeConstrainedEnabled()

RTI 1.3-NG

ABSTRACT

This callback advises the federate that time constraint has been enabled, as per a previous `enableTimeConstrained()` service invocation.

HLA IF SPECIFICATION

This method realizes the "Time Constrained Enabled" Time Management service as specified in the *HLA Interface Specification* (§8.6 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   timeConstrainedEnabled (
      const RTI::FedTime& theFederateTime
   )
throw (
   RTI::InvalidFederationTime,
   RTI::EnableTimeConstrainedWasNotPending,
   RTI::FederateInternalError
)
```

ARGUMENTS

theFederateTime

the logical time of the federate at which time constraint takes effect

DESCRIPTION

This callback advises the federate of the successful completion of an `enableTimeConstrained()` service invocation. Upon such a callback, the advantages and limitations associated with time constraint become effective.

The logical time argument to this callback will be the logical time of the federate at the point in execution at which the federate requested time constraint.

RTI 1.3 does not require any negotiation among federates to enable time constraint, so upon a successful `enableTimeConstrained()` invocation, a `timeConstrainedEnabled()` callback will be immediately scheduled for subsequent delivery during an invocation of `tick()`.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges that time constraint has taken effect.

EXCEPTIONS

RTI::InvalidFederationTime

The specified grant time is invalid.

RTI::EnableTimeConstrainedWasNotPending

An `enableTimeConstrained()` request is not currently outstanding for the local federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FedTime
RTI::RTIambassador::
enableTimeConstrained()

B.5.4 timeRegulationEnabled()

RTI 1.3-NG

ABSTRACT

This callback advises the federate that time regulation has been enabled per a previous `enableTimeRegulation()` service invocation.

HLA IF SPECIFICATION

This method realizes the "Time Regulation Enabled" Time Management service as specified in the *HLA Interface Specification* (§8.3 in version 1.3).

SYNOPSIS

```
#include <RTI.hh>

virtual
void
RTI::FederateAmbassador::
   timeRegulationEnabled (
      const RTI::FedTime& theFederateTime
   )
throw (
   RTI::InvalidFederationTime,
   RTI::EnableTimeRegulationWasNotPending,
   RTI::FederateInternalError
)
```

ARGUMENTS

theFederateTime

the logical time at which regulation as been enabled

DESCRIPTION

This callback advises the federate of the successful completion of an `enableTimeRegulation()` service invocation. Upon such a callback, the advantages and limitations associated with time regulation become effective.

The federate's logical time upon enabling time regulation will be the minimum of the *effective* federate logical time specified to the `enableTimeRegulation()` service invocation and the current lower-bound time-stamp of the federation.

RTI 1.3 does not require any negotiation among federates to enable time regulation, so upon a successful `enableTimeRegulation()` invocation, a `timeRegulationEnabled()` callback will be immediately scheduled for subsequent delivery during an invocation of `tick()`.

RETURN VALUES

A non-exceptional return indicates that the federate acknowledges that time regulation has taken effect.

EXCEPTIONS

RTI::InvalidFederationTime

The specified logical time argument does not represent a valid point on the federation time axis.

RTI::EnableTimeRegulationWasNotPending

An `enableTimeRegulation()` request is not currently outstanding for the local federate.

RTI::FederateInternalError

An error internal to the federate has occurred.

SEE ALSO

RTI::FedTime
RTI::RTIambassador::
enableTimeRegulation()