High Level Architecture
Module 1
Basic Concepts
High Level Architecture
Module 1
Basic Concepts

Part 5
Declaration Management

• Declaration Management serves the purpose of coordinating data exchange between federates.

• Declaration Management:
  – Specifies the data a federate will send and receive.
  – Controls where data is sent based on external interest.
Declaration Management

- Declaration Management includes publication, subscription, and supporting control functions.
  - Federates that produce objects (or object parts) or that produce interactions must declare exactly what they are able to publish (i.e. generate and provide to the federation.)
  - To begin receiving updates of an attribute or an interaction, a federate must declare (subscribe) its interest in the attribute.
The RTI uses this information to match what is being published to information which has been subscribed to by other federates. This matching enables the RTI to tell producing federates whether to bother updating an attribute or producing interactions of a given class.
Object Vocabulary

- **Object Classes**
  - Comprised of Attributes
  - Describe Types of Things That Can **Persist**

- **Interaction Classes**
  - Comprised of Parameters
  - Describe Types of Events (Transitory)

- **Objects Persist, Interactions Do Not**
Class Hierarchy - Venn Diagram

W: \{a, b, c, d\}

X: \{a, b, c, d, e, f, g\}

Z: \{a, b, c, d, i, j\}

Y: \{a, b, c, d, e, f, g, h\}

Taken from the Hands On Practicum
Object Publication And Subscription

Federate #1 Publishes Y with the following attributes: \{b, e, f, g, h\}.

Federate #2 Subscribes to X with the following attributes: \{a, b, c, d, e\}. 
Interaction Publication

- As with object classes, each federate must state explicitly which interaction classes it intends to produce.
- Interactions are produced as “all or nothing”.
  - It isn’t possible to specify which parameters in an interaction will be published.
Object Subscription and Type Promotion

- If subscribed to W and X:
  - Instances of Class W and X seen without promotion
  - Instances of Class Y seen as instances of Class X
  - Instances of Class Z seen as instances of Class W
Interaction Subscription

• Each federate subscribes to the interaction classes it wishes to receive.

• It is not possible to subscribe to individual parameters of an interaction class.

• A federate is informed about a new interaction instance if
  – The federate has subscribed to the interaction class of the instance
  – The instance can be promoted (i.e., up the hierarchy) to a subscribed interaction class.
Declaration Management
Objects

White Federate
Fed. Code: FederateAmbassador
LRC: RTIAmbassador

- getObjectClassHandle() A.7.20
- getAttributeHandle() A.7.11
- publishObjectClass() A.2.2
- startRegistrationForObjectClass() B.2.2
- stopRegistrationForObjectClass() B.2.5
- unpublishObjectClass() A.2.11

Green Federate
Fed. Code: FederateAmbassador
LRC: RTIAmbassador

- subscribeObjectClassAttributes() A.2.6
- unsubscribeObjectClass() A.2.14

Taken from the Hands On Practicum
Declaration Management
Interactions

White Federate
Fed. Code: FederateAmbassador
LRC: RTIAmbassador

Green Federate
Fed. Code: FederateAmbassador
LRC: RTIAmbassador

- getInteractionClassHandle()  A.7.16
- publishInteractionClass()  A.2.1
- turnInteractionsOn()  B.2.8
- turnInteractionsOff()  B.2.7
- unpublishInteractionClass()  A.2.10
- subscribeInteractionClass()  A.2.4
- unsubscribeInteractionClass()  A.2.13

California State University, Chico
Taken from the Hands On Practicum
3/13/99
Objects

- To create an object, the federate must have published that object class (declaration mgt.)
- The federate registers the new instance of the object
- To discover this object, other federates must have subscribed to that object class (declaration mgt.)
- Those federates then discover the object instance
Object Management Methodology

White Federate
Fed. Code: FederateAmbassador
LRC: RTIAmbassador
- registerObjectInstance() A.3.9
- turnUpdatesOnForObjectInstance() B.3.13
- deleteObjectInstance() A.3.5

Green Federate
Fed. Code: FederateAmbassador
LRC: RTIAmbassador
- discoverObjectInstance() B.3.4
- removeObjectInstance() B.3.12
Object Management Updates

White Federate
Fed. Code: FederateAmbassador
LRC: RTIAmbassador

provideAttributeValueUpdate()
updateAttributeValues()

Green Federate
Fed. Code: FederateAmbassador
LRC: RTIAmbassador

requestClassAttributeValueUpdate()
requestObjectAttributeValueUpdate()

reflectAttributeValueUpdate()
B.3.8
A.3.11
A.3.13
B.3.5
A.3.16

Taken from the Hands On Practicum
California State University, Chico
3/13/99
Interactions

• To send an interaction, the federate must have published that interaction class (per declaration management)

• To receive this interaction, other federates must have subscribed to that interaction class (per declaration management)
Object Management Interactions

White Federate
Fed. Code: FederateAmbassador
LRC: RTIAmbassador

sendInteraction() \rightarrow A.3.14

Green Federate
Fed. Code: FederateAmbassador
LRC: RTIAmbassador

receiveInteraction() \rightarrow B.3.6
Additional Object Control

Federate Initiated Controls

changeAttributeTransportType (RTI::ObjectHandle, const RTI::AttributeHandleSet&, RTI::TransportationHandle)

changeInteractionTransportType (RTI::InteractionClassHandle, RTI::TransportationHandle)

RTI Initiated Controls

turnUpdatesOnForObjectInstance (RTI::ObjectHandle, const RTI::AttributeHandleSet&)

turnUpdatesOffForObjectInstance (RTI::ObjectHandle, const RTI::AttributeHandleSet&)

user_code

fedamb : Federate Ambassador

rtiamb : RTIambassador

various : Misc RtiLibClasses
Scope Interactions

received after object is registered indicating federate is clear to provide attribute updates.

enableAttributeScopeAdvisorySwitch ()

attributesInScope (RTI::ObjectHandle, const RTI::AttributeHandleSet&)

attributesOutOfScope (RTI::ObjectHandle, const RTI::AttributeHandleSet&)

disableAttributeScopeAdvisorySwitch ()