«interface» Messages::Message Configuration

+ createForwardingNode(newNetworkNode): void
+ createNewMessageCategory(): void
+ defineNewComplexMessageMetadataFieldType(fieldTypeName, fieldTypeDescription, selectedMetadataTypeSpec): void
+ defineNewMessageType(messageTypeDetails): void
+ defineNewMetadataTypeSpecification(metadataTypeSpecDetails): void
+ defineNewSimpleMessageMetadataType(fieldTypeName, fieldTypeDescription): void
+ deprecateMessageTypes({MessageTypesToDeprecate}): void
+ getMessageCategories(): {MessageCategory}
+ getMessageMetadataFieldTypeDetails(metadataFieldTypeName): void
+ getMessageTypeDetails(messageName): MessageTypeDetails
+ getMessageTypes(): {MessageTypes}
+ getMetadataFieldTypes(): {Message Metadata Field Type}
+ getMetadataTypeSpecificationDetails(metadataTypeName): MetadataTypeSpecificationDetails
+ getMetadataTypeSpecifications(): {Metadata Type Specification}
+ makeMessageTypesReadyForUse({MessagesToBeMadeReadyForUse}, readinessType): void
+ updateComplexMessageMetadataFieldType(updatedComplexMessageMetadataFieldType): void
+ updateMessageType(updatedMessageTypeDetails): void
+ updateMetadataTypeSpecification(updatedMetadataTypeSpecificationDetails): void
+ updateSimpleMessageMetadataFieldType(updatedSimpleMessageMetadataFieldType): void

«interface» Messages::Message Search

+ findMessages(searchCriteria): void
+ getMessages(MessageList): void

«interface» Messages::Message Logging

+ logSystemGeneratedMessage(conversationID, messageTypeName, messageDescription, messageMetadataFieldSequence): void
+ logUserGeneratedMessage(conversationID, messageTypeName, messageDescription, messageMetadataFieldSequence, messageInitiator): void
+ receiveForwardedMessage(forwardedMessage): void

«interface» Messages::Message Subscriptions

+ addSubscriptions(subscriberID, {messageTypesForSubscription}): void
+ getMessageSubscriptionsForSubscriber(subscriber): {Subscribed Message Types}
+ removeSubscriptions(subscriberID, {messageTypesForUnsubscription}): void

«todo»
Consider whether we could merge the two log message operations into one if both can have an initiator (which is currently limited to only manually generated messages)

«todo»
Also, consider changing the parameters of the two (or one if they are merged) 'logMessage' operations so that the signature uses the newly created data type MessageDetails instead of the individual parameters.

Logged Messages	
#	**Analytics-Generated Messages**
1	New Analytic Installed
2	Analytic activated for Field-Trial Use Only
3	Analytic activated for Full Use
4	Schedule Changed for Scheduled Analytic
5	Analytic Deactivated
6	Analytic Deleted
7	Scheduled Analytic Execution Started
8	Scheduled Analytic Execution Completed
9	Start of User-Invoked Analytic Execution
10	Completion of User-Invoked Analytic Execution
11	Analytic Restored
12	Analytic Result Retrieved By User
13	Analytic Uninstalled
14	Result Recipients Added for Scheduled Analytic
15	Result Recipients Removed for Scheduled An...
	...more to come!
#	**Messages-Generated Messages**
1	New Message Type Created
2	New Message Metadata Field Type Created
3	New Metadata Type Specification Created
4	Message Type Updated
5	Message Metadata Field Type Updated
6	Metadata Type Specification Updated
7	Forwarding Nodes For Message Type Modified
8	Message Type Made Ready for Trial Use
9	Message Type Made Ready for Full Use
10	Message Type Deprecated
11	New subscription set up
12	Subscription removed
13	New Message Logged
	...more to come

Messages

«interface»
Messages::Notifications

+ reportConversation(): void
+ reportMessage(): void

Notifications

Message
Forwarding

«interface»
Messages::Message Forwarding

+ forwardMessage(forwardedMessage, forwardingNode): void

«todo»
Given that it is possible to validate
the message metadata field
sequence for a logged message it
might make sense for the
logMessage operation to return
an indication of message validity