This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-0982

Cyber Effects Simulation Interface v1.6.6

*Note: This document is a work-in-progress and is being provided only to inform related development efforts, and not as a candidate release. Additional changes **will** occur before release. The current test example is attached here:*

Testing API.postman_collection.json

The Cyber Effects Simulation Interface provides a way to stimulate message communication between a kinetic based simulation and a cyber simulation.

Route:	CKI/v1.6/DataTypes
Headers:	No explicit headers needed
Parameters:	No parameters used
Methods:	GET, POST
Body:	JSON (schema below)

GET: Returns an array of known data types.

Response Body:

```
{
     "dataTypes {
          "simulationID" : (Name)
          "supportedDataTypes" : [
               {"Name": (Data Type Name), "ID": (Integer), "UUID":
          (UUID)}, ...(for each data type) ]
          }
}
```

Field	Description
simulationID	Name of the simulation publishing the message.
supportedDataTypes	An array of data type objects consisting of name/ID number/guid triples.

Return Value	Description
200 OK	Returned body includes valid list of data types
400 Bad Request	Request was not understood

POST: Notifies simulation of known data type for federate

Request Body:

```
{
     "dataTypePublication {
          "simulationID" : (Name)
          "dataTypeName" : (   Data Type Name)
          "UUID" : ( UUID )
          "attribute" : {
               "attributeName" : (Name)
               "attributeType" : (bool|int|double|complex type|etc)
               "confidence" : { "canonical sim": (name), "value" :(0-
100) }
          }
          ….(for each attribute)
     }
}
```

Field	Description
aimulationID	Name of the simulation publishing the message.
dataTypeName	The name of the data type being published.
attribute	The attribute description object contains nested fields to define the kind of attribute it is.
attributeName	The name of the attribute.
attributeType	This field defines the data type of the attribute. It can be a primitive data type (boolean, integer, double, float, etc), or it can be a complex type. In the case of complex types, the field will be represented by a nested object composed of multiple attributes.
confidence	This is the level of confidence the publishing simulation is reporting about it's ability to simulate and set this attribute (or the sim it considers canonical).

Return Value	Description
200 OK	Notification received but not supported. (IE: Ignored. I don't model this)
201 Created	Notification received and supported (IE: I can model this)
400 Bad Request	Request was not understood
500 Server Error	Conflicting Data type exists (Same GUID but different Schema. Fatal error).

Route:	CKI/v1.6/DataTypes/N *(Where N is an integer)*
Headers:	No explicit headers needed
Parameters:	No parameters used
Methods:	GET
Body:	JSON (schema below)

GET: Returns full details of the data type with ID = N.

Response Body:

```
{
      "dataTypePublication {
            "simulationID" : (Name)
            "dataTypeName" : (Data Type Name)
            "attributes" : [
            {
                  "attributeName" : (Name)
                  "attributeType" : (bool|int|double|complex type|etc)
                  "confidence" : (0-100)
            },
            {
                  "attributeName" : (Name)
                  "attributeType" : (bool|int|double|complex type|etc)
                  "confidence" : (0-100)
            }
            ….(for each attribute)]
      }
}
```

Field	Description
simulationID	Name of the simulation publishing the message.
dataTypeName	The name of the data type being published.
attribute	The attribute description object contains nested fields to define the kind of attribute.
attributeName	The name of the attribute.
attributeType	This field defines the data type of the attribute. It can be a primitive data type (boolean, integer, double, float, etc), or a complex type. In the case of complex types, the field will contain a nested object composed of multiple attributes.
confidence	This is the level of confidence the publishing simulation is reporting about it's ability to simulate and set this attribute.

Return Value	Description
200 OK	Body contains detailed information about this type of system.
400 Bad Request	Request was not understood
404 Not Found	Requested data type does not exist

Route:	CKI/v1.6/DataTypes/N/Systems *(Where N is an integer)*
Headers:	No explicit headers needed
Parameters:	No parameters used NOTE: What search params should we support?
Methods:	GET, POST
Body:	JSON (schema below)

GET: Returns a list of all systems of type N.

Response Body:

{

"***systemList {***

"systems" : [{"name" : (name), "ID" : (ID)}, ...]

}

}

Field	**Description**
systems	Contains an array of objects consisting of system name/id pairs.

* Note: We will want to allow for additional filtering, maybe a search param where attribute values can be checked to allow filtering?

Return Value	**Description**
200 OK	Body contains list of systems matching the given type.
400 Bad Request	Request was not understood

POST: Notify of creation of new system within federate's simulation.

Request Body:

```
{
     "systemCreated": {
          "simulationID" : (Name, UUID)
          "timeStamp" : (Time)
          "Name": (Human Readable Name)
          "systemID": (UUID)
          "urn"      : (URN if the device has a URN)
          "systemState" : {
               Body of specific system type (See System State Body)
          }
     }
}
```

Field	**Description**
simulationID	Name of the simulation publishing the response message.
timeStamp	The time the message was published. a string representing the date and time in ISO 8601 format, YYYY-MM-DDTHH:MM:SS
systemID	The Unique Identifier for the instance of the system being published.
Urn	If the system being published has been assigned a URN, this attribute will be published. Otherwise, this field is left out.
systemState	This field represents the embedded SystemState object that contains state information relevant to the system being published.

Return Value	**Description**
200 OK	Confirmed creation but will not model locally. (No error, but ignored)
201 Created	Body contains ID of matching system created in this sim.
400 Bad Request	Request was not understood
404 Not Found	That type of system doesn't exist here!

Route:	CKI/v1.6/DataTypes/N/Systems/M *(Where N and M are integers)*
Headers:	No explicit headers needed
Parameters:	No parameters used
Methods:	GET, PUT, POST, DELETE
Body:	JSON (schema below)

GET: Returns all current simulation values for system M of the type N

Response Body:

```
{
        "SystemStateUpdate": {
                "SimulationID" : (Name, UUID)
                "TimeStamp" : (Time)
                "SystemID": (Name, UUID,)
                "URN"          : (URN if the device has a URN)
                "SystemState" : {
                        Body of specific system type (See System State Body)
                }
        }
}
```

Field	**Description**
simulationID	Name of the simulation publishing the response message.
timeStamp	The time the message was published. a string representing the date and time in ISO 8601 format, YYYY-MM-DDTHH:MM:SS
systemID	The Unique Identifier for the instance of the system being published.
urn	If the system being published has been assigned a URN, this attribute will be published. Otherwise, this field is left out.
systemState	This field represents the embedded SystemState object that contains state information relevant to the system being published. Note, in the System Update message, the ***Confidence*** field is omitted.

Return Value	**Description**
200 OK	Body contains system information.
400 Bad Request	Request was not understood

PATCH: Set given value for system M of type N.

Body:

```
{
        "SystemStateUpdate": {
                "SimulationID" :
                "TimeStamp" : (Time)
                "SystemID": (Name, UUID, etc)
                "URN"          : (URN if the device has a URN)
                "SystemState" : {
                        Body of specific system type (See System State Body)
                }
        }
}
```

Field	**Description**
simulationID	Name of the simulation publishing the response message.
timeStamp	The time the message was published. a string representing the date and time in ISO 8601 format, YYYY-MM-DDTHH:MM:SS
systemID	The Unique Identifier for the instance of the system being published.
urn	If the system being published has been assigned a URN, this attribute will be published. Otherwise, this field is left out.
systemState	This field represents the embedded SystemState object that contains state information relevant to the system being published. Note, in the System Update message, the ***Confidence*** field is omitted.

Return Value	**Description**
200 OK	Body contains system information.
400 Bad Request	Request was not understood
403 Forbidden	Attempting to modify attribute this system does not know you as canonical for.
404 Not Found	Provided System ID does not exist.

POST: Send message to system M of type N.

Body:

```
{
     "MessageInteraction": {
          "SimulationID" : (Name)
          "Sender" : (System ID)
          "TimeStamp" : (Time)
          "MessageType": (Binary|Tactical|Metadata)
          "MessageBody" : {
               (Message Payload)
          }
     }
}
```

Field	**Description**
simulationID	The name of the simulation from which this interaction originated.
sender	The instance ID of the system that is sending the tactical message.
recipients	An array of instance IDs of one or more systems that are the recipient of this tactical message.
timeStamp	The time the message was published. a string representing the date and time in ISO 8601 format, YYYY-MM-DDTHH:MM:SS
messageType	The message type. This field identifies which message type is found in the Message Body
messageBody	A complex object that represents the contents of the actual message. This object will be base 64 encoded for a Binary message type, a JSON object for Metadata data type, and for a Tactical message type will include a pair of the message type and the message data.

Return Value	**Description**
200 OK	Body contains system information.
400 Bad Request	Request was not understood

DELETE: Notify of deletion of system M of type N.

Body:

```
{
     "SystemRemoved": {
          "SimulationID" : (Name)
          "TimeStamp" : (Time)
          "SystemID": (Name, UUID, etc)
          "URN"     : (URN if the device has a URN)
     }
}
```

Field	**Description**
simulationID	Name of the simulation publishing the response message.
timeStamp	The time the message was published. a string representing the date and time in ISO 8601 format, YYYY-MM-DDTHH:MM:SS
systemID	The Unique Identifier for the instance of the system being published.
urn	If the system being published has been assigned a URN, this attribute will be published. Otherwise, this field is left out.

Return Value	**Description**
204 No Content	System deleted from server's simulation
304 Not Modified	Deletion notification received, but system not removed from server's simulation
400 Bad Request	Request was not understood

Route:	CKI/v1.6/DataTypes/N/Systems/M/Subscriptions *(Where N and M are integers)*
Headers:	No explicit headers needed
Parameters:	No parameters used
Methods:	GET, POST
Body:	JSON (schema below)

GET: Returns all current subscriptions for this system

Response Body:

{
"Subscriptions": {
[{"ID" : (ID), "SystemID" : (Name, UUID), attibutes : { "AttributeName": (attribute name), … } , Target: (TargetURL)}]
}
}
}

Field	**Description**
subscriptions	Contains an array of subscription objects
systemID	Identification of the a system being simulated.
~~subscriber~~	~~Identification of the federate that needs to be informed of changes to the state of the system being simulated.~~
attributes	Array of all the attributes for which subscriber will be notified of updates.
target	URL to which updates will be posted

Return Value	**Description**
200 OK	Body contains subscription information.
400 Bad Request	Request was not understood

POST: Creates a new subscription

Request Body:

```
{
        "SubscriptionRequest": {
                "SystemID" : (Name, UUID),
                "Attributes": {},
                "Target": (TargetURL)
        }

}
```

Field	**Description**
systemID	Identification of the a system being simulated.
~~subscriber~~	Identification of the federate that needs to be informed of changes to the state of the system being simulated.
attributeName	Array of all the attributes for which subscriber will be notified of updates.
target	URL to which updates will be posted

Return Value	**Description**
201 Created	New subscription created (Return Body will contain ID of new subscription)
400 Bad Request	Request was not understood
404 Not Found	Requested subscription for content this server is not canonical for / doesn't simulate

NOTE: Once a subscription is created, the server will PATCH updates to the 'targetURL' path in the format of:

```
        "SystemStateUpdate": {
                "SimulationID" : (Name, UUID)
                "TimeStamp" : (Time)
                "SystemID": (Name, UUID,)
                "URN"          : (URN if the device has a URN)
                "SystemState" : {
                        Body of specific system type (See System State Body)
                }
        }
```

Note that this is identical to the `CKI/v1.6/DataTypes/N/Systems/M` GET request response structure. (And should remain so if updates are made to this document.

Route:	CKI/v1.6/DataTypes/N/Systems/M/Subscriptions/X *(Where N,M and X are integers)*
Headers:	No explicit headers needed
Parameters:	No parameters used
Methods:	GET, PUT, DELETE
Body:	JSON (schema below)

GET: Returns a specific subscription

Response Body:

{
"Subscription": {
{"SystemID" : (Name, UUID), Attributes: { "AttributeName": (attribute name), … } , Target: (TargetURL) }
}
}

Field	**Description**
systemID	Identification of the a system being simulated.
~~subscriber~~	~~Identification of the federate that needs to be informed of changes to the state of the system being simulated.~~
attributeName	Array of all the attributes for which subscriber will be notified of updates.
target	URL to which updates will be posted

Return Value	**Description**
200 OK	Body contains subscription information.
400 Bad Request	Request was not understood
404 Not Found	Requested subscription does not exist (bad ID)

PUT: Update a subscription (Note, replaces previous subscription)

Request Body:

{
"SubscriptionRequest": {
[{"SystemID" : (Name, UUID), "Subscriber": (Simulation ID), ["AttributeName": (attribute name), …] , Target: (TargetURL)}]
}
}
}

Field	**Description**
systemID	Identification of the a system being simulated.
subscriber	Identification of the federate that needs to be informed of changes to the state of the system being simulated.
attributeName	Array of all the attributes for which subscriber will be notified of updates.
target	URL to which updates will be posted

Return Value	**Description**
201 Created	New subscription created
400 Bad Request	Request was not understood
404 Not Found	Requested subscription for content this server is not canonical for / doesn't simulate or ID doesn't exist.

Delete: Removes a subscription

Request Body:
(none)

Return Value	Description
204 No Content	Sucessfully deleted
400 Bad Request	Request was not understood
404 Not Found	Requested subscription doesn't exist

System State

The system state represents the state of each type of system. The types of attributes can vary between system types. For example, a SCADA system may have different attributes than a GPS Receiver system. The format of each body is agreed upon between simulations during the publication phase. The System State Body is NOT a message in itself; it is included as part of the ***System Created and System State Update*** interactions as part of the payload. Only attributes that are being set will be populated. For example, at runtime, the confidence value will be absent because simulation-attribute ownership was determined prior to scenario start.

System State Body Template:

```
{
    "SystemState":{
        "Attribute" : [{
            "AttributeName": (attribute name)
            "Value" : (attribute value)
        },
        "Attribute" : [{
            "AttributeName": (attribute name)
            "Value" : (attribute value)
        }
        …(for each attribute)
        ]
    }
}
```

Field	Description
dataTypeName	Name of the data type that is represented by this JSON body.
attributeName	The name of the attribute. This is a nested object that contains additional values to make up the attribute.
value	The value of the attribute being set. If the value is a complex type, this will be a nested object of values.
confidence	This is the level of confidence the publishing simulation is reporting about it's ability to simulate and set this attribute.