ENTERPRISE Data Service Technical Reference Guide

Version 2020

Change History

This table shows the change history of this guide:

Edition	Date	Reason
1	05 February 2020	First edition.

Table of Contents

1 About the ENTERPRISE Data Service

The ENTERPRISE Data Service (EDS) is a web service which provides several interfaces for accessing and manipulating data from a pre-existing ENTERPRISE deployment. Its architecture consists of two middle-tier service layers which are able to access a single ENTERPRISE database:

Presentation Service – This service is responsible for communicating with the outside world and offers either a traditional web service interface or a convenient REST interface for prototyping queries or diagnosing faults.

Persistence Service – This service is responsible for processing work items from the presentation service and interacting with the back-end Oracle database. The persistence service may be clustered if large concurrent access to the service is required, but it should be noted that each instance will create a 10 user connection pool to the Oracle server.

The two services are attached to each other using a duplexed pair of message queues which allows for either single server or multiple server deployments.

Notes:

- A single server configuration is the default setting, but you are still able to run concurrent persistent instances from the same server as long as they are attached to the same presentation service.
- For software building purposes EDS 2020 is designated as version 10.0 in some contexts.

ENTERPRISE Data Types Supported Within the EDS

EDS data types map on to equivalent generic ENTERPRISE data types. The major supported ENTERPRISE data types are listed in the following set of tables. These tables incorporate all the currently supported ENTERPRISE data types, including those supported by the newly added EDS data types in this Release. For a separate table showing the newly added EDS data types, see New Data Types Added to this Release on page 13.

Note: For the complete list of EDS data types, including those which have extensions from previous releases, we recommend that you use the REST interface:

```
        http://<server>:8733/Aircom/EDS/REST/
```

This table lists the major supported ENTERPRISE data types general to all products (ASSET and ASSET Backhaul*):

ENTERPRISE Type	CRUD Support	Technology	Category	Product
Project	R	-	Config	All
Location Object (Property)	CRUD	-	Element	All
Location Object Hierarchy	R	-	Element	All
Feeder	CRU	-	Equipment	All
Propagation Model	R	-	Config	All

(* Prior to 9.1, ASSET Backhaul was known as CONNECT.)

This table lists the major supported ENTERPRISE data types specific to ASSET:

ENTERPRISE Type	CRUD Support	Technology	Category	Product
Propagation Model - see table above				
Cellular Antenna	CRU	-	Equipment	ASSET
BTS Equipment	R	GSM	Equipment	ASSET
GSM Sub Cell Layer Definition (see Note1)	CRU	GSM	Config	ASSET
GSM Carrier Layer Definition (see Note1)	CRU	GSM	Config	ASSET
GSM Carrier Definition (see Note1)	CRU	GSM	Config	ASSET
MSC (see Note1)	CRUD	GSM	Element	ASSET
BSC (see Note1)	CRUD	GSM	Element	ASSET
Cell (see Note 5)	CRUD	All	Element	ASSET
CellSite	CRUD	GSM	Element	ASSET
GSM Cell (see Note 2)	CRUD	GSM	Element	ASSET
GSM Repeater	CRUD	GSM	Element	ASSET
GSM Subcell	RU	GSM	Element	ASSET
WMSC	CRUD	UMTS	Element	ASSET
RNC	CRUD	UMTS	Element	ASSET
SGSN	CRUD	UMTS	Element	ASSET
PLMN Network	CRU	UMTS	Element	ASSET
NodeB (see Note 3)	CRUD	UMTS	Element	ASSET
UMTS Carrier Definition	RU	UMTS	Config	ASSET
UMTS Cell (see Note 4)	CRUD	UMTS	Element	ASSET
UMTS Repeater	CRUD	UMTS	Element	ASSET
Network Connection	CRUD	All	Element	ASSET
Logical Network Connection	CRUD	All	Element	ASSET
Cellular Network Connection	CRUD	All	Element	ASSET
LTE Node	CRUD	LTE	Element	ASSET
LTE Cell (see Note 4)	CRUD	LTE	Element	ASSET
LTE Repeater	CRUD	LTE	Element	ASSET
LTE MME	CRUD	LTE	Element	ASSET
LTE Carrier Definition	RU	LTE	Config	ASSET
LTE PF Scheduler MUG	R	LTE	Config	ASSET
Wi-Fi Node	CRUD	Wi-Fi	Element	ASSET
Wi-Fi Cell (see Note 4)	CRUD	Wi-Fi	Element	ASSET
Wi-Fi Carrier Definition	RU	Wi-Fi	Config	ASSET
Terminal Types	R	All	Config	ASSET
Bearers	R	GSM/UMTS/LTE	Config	ASSET
Services	R	All	Config	ASSET
MU-Node	CRUD	All	Element	ASSET
LTE eICICPattern	CRU	LTE	Config	ASSET
LTE Multipath Channel Model	RU	LTE	Config	ASSET
Neighbour Limit	R	-	Config	ASSET

Notes:

- Delete support is deliberately prohibited on MSC, BSC and other indicated elements to prohibit accidental removal of any attached GSM sites or configuration data. This restriction will be removed in future release once EDS implements role based security.
- As part of their definition, GSM Cells include Sub-Cells, Carrier Allocations, Antennas and other equipment assignments. Refer to WSDL for more detail.
- As part of their definition, NodeBs include UMTS Carrier assignments, Antennas and other equipment assignments. Refer to WSDL for more detail.
- As part of their definition, UMTS, LTE and Wi-Fi Cells include carrier, antenna and feeder assignments, as well as other parameters. Refer to WSDL for more detail.
- This corresponds to the description of the major changes in this release, described in Special Changes in this Release on page 15. The ‘Cell’ in version 10.0 represents the ‘multi-tech’ cell, which means that the cell object now has the potential to be configured for multiple technologies. It also relates to why there is no separate cell object for the new 5G technology (unlike the other technologies which retain their technology-specific data types in EDS for backward compatibility with 9.1).

This table lists the major supported data types specific to ASSET Backhaul*:

ENTERPRISE Type	CRUD Support	Technology	Category	Product
MW Antenna	CRU	-	Equipment	ASSET Backhaul
Radio Equipment	CRU	-	Equipment	ASSET Backhaul
Band	CRU	-	Config	ASSET Backhaul
Point to Point Link	CRUD	-	Element	ASSET Backhaul
Frequency Band	R	-	Config	ASSET Backhaul
Back To Back Passive Repeater	CRUD	-	Element	ASSET Backhaul
Dual Polar Link	CRUD	-	Element	ASSET Backhaul
Dual Polar Sub Link	CRUD	-	Element	ASSET Backhaul
Multi Radio Link	CRUD	-	Element	ASSET Backhaul
Multi Radio Sub Link	CRUD	-	Element	ASSET Backhaul
Point to Multi Point Carrier	CRUD	-	Element	ASSET Backhaul
Point to Multi Point Hub	CRUD	-	Element	ASSET Backhaul
Point to Multi Point Link	CRUD	-	Element	ASSET Backhaul
Point to Multi Point Sector	CRUD	-	Element	ASSET Backhaul
Reflector Passive Repeater	CRUD	-	Element	ASSET Backhaul
Reflector Repeater	CRUD	-	Element	ASSET Backhaul
Modulation Type	CRU	-	Config	ASSET Backhaul
T/I Objectives	CRU	-	Config	ASSET Backhaul

(* Prior to 9.1, ASSET Backhaul was known as CONNECT.)

Backwards Compatibility with Previously Released Data Types

There are some significant changes in this release:

For more information, see Special Changes in this Release on page 15.

There are some newly added data types for this release. See New Data Types Added to this Release on page 13.

The following table may help to clarify the behaviour of backwards compatibility:

Operation	Element	EDS	Behaviour
Read	Node (Any)	< 9.1	Backwards compatibility will only show Logical Antennas that are attached to Cells owned by Node (for GSM, see next row). Only Feeders that are attached to Cells located on the same Property as the Node will be visible. Only Cells that are co-located on the same Property as the Node will be visible. It will still be possible to see all cells via the explicit EDS cell type.
Read	Cell (GSM)	< 9.1	Backwards compatibility will only show Logical Antennas that are attached to Cell Layers. Only Logical Antennas that are co-located on the same Property as the GSM Cell will be visible (to prevent duplicate lists).
Create/ Update	Cell (Any)	< 9.1	Cell location will be the same as the parent node (MU-Node). New GSM Logical Antennas are permitted as long as the Logical Index is unique to the Property of that Cell. Updates to Logical Antennas are allowed and will be applied to the appropriate Property.
Replace	Logical Antennas/ Feeders (Any)	< 9.1	Where EDS has been instructed to replace an Antenna or Feeder list it will need to filter out any elements which are not co-located on the same Property as the Node (or GSM cell). This is to protect the database from accidental deletion of remote Antennas/Feeders by legacy EDS types. EDS will not be allowed to delete Logical Antennas via a legacy Node or Cell type. Where a reference to a Cell Feeder is removed, it will be detached from the Antenna, but the Logical Antenna will not be affected.
Delete	Cell (Any)	< 9.1	Cell deletion should behave almost the same as it has done previously. EDS will detach any references to remote Logical Antennas. GSM Cell deletion will remove cellular references to Logical Antennas, but it will not delete remote Logical Antennas from the Property.
Replace	Cell (Any)	< 9.1	Where EDS has been instructed to replace a Cell list owned by a given Node/CellSite, it will need to filter out any Cells which are not co-located on the same Property as the Node (or GSM cell). This is to protect the database from accidental deletion of remote Cells by legacy EDS types.
Create/ Update	Cell (Any)	9.1	EDS will validate the specified Property location for existence. If it is not specified, the default will be the same as for the parent node.
Create/ Update	Node (Any)	<= 9.1	Node location functionality should remain unaffected. Any newly created cells beneath the Node will implicitly share the same location as the Node. New Logical Antennas are permitted as long as the Logical Index is unique to the Property of that Node.

Operation	Element	EDS	Behaviour
			For Repeaters in 10.2 all Repeaters by default are supported by all technologies. So when a Repeater is created in 10.2, EDS will only be able to return a v100 type. If a Repeater was created in an older version of Asset and has been upgraded, then the SUBTYPEFK column in the LOGNODE table is set to the original ObjectType, therefore allowing EDS to emulate the original technology for all supported versioned types.

New Data Types Added to this Release

This table summarises the new EDS data types that have been added to the version 10.0 release:

EDS Type	CRUD Support	Technology	Category	Product
AntennaDevicev100Type	CRU	-	Element	ASSET
AntennaDevicev103Type	CRU	-	Element	ASSET
AntennaPatternv100Type	CRU	-	Element	ASSET
AntennaPatternv103Type	CRU	-	Element	ASSET
Cellv100Type	CRUD	All	Element	ASSET
Cellv103Type	CRUD	All	Element	ASSET
Cellv104Type	CRUD	All	Element	ASSET
LocationObjectHierarchyv100Type	R	-	Element	All
LocationObjectHierarchyv103Type	R	-	Element	All
LocationObjectHierarchyv104Type	R	-	Element	All
LocationObjectv100Type	CRUD		Element	All
LocationObjectv103Type	CRUD		Element	All
LocationObjectv104Type	CRUD	-	Element	All
LTECarrierv100Type	RU	LTE	Config	ASSET
MUNodev100Type	CRUD	All	Element	ASSET
MUNodev103Type	CRUD	All	Element	ASSET
MUNodev104Type	CRUD	All	Element	ASSET
NeighbourType	RUD	-	Element	ASSET
Neighbourv100Type	RUD	-	Element	ASSET
NRAttenuationFilterv103Type	CRU	5G	Config	ASSET
NRBearerv100Type	R	5G	Config	ASSET
NRCarrierv100Type	RU	5G	Config	ASSET
NRCodeSchemav100Type	R	5G	Config	ASSET
NRFramev100Type	R	5G	Config	ASSET
NRFreqBandv100Type	R	5G	Config	ASSET
NRRSISchemav103Type	R	5G	Config	ASSET

EDS Type	CRUD Support	Technology	Category	Product
Projectv104Type	CRUD*	-	Element	ASSET
Repeaterv100Type	CRUD	All	Element	ASSET
ServiceMultiTechv100Type	R	All	Config	ASSET
ServiceMultiTechv104Type	R	All	Config	ASSET
Servicev100Type	R	All	Config	ASSET
Servicev104Type	R	All	Config	ASSET
TerminalTypeMultiTechv100Type	R	All	Config	ASSET
TerminalTypeMultiTechv103Type	R	All	Config	ASSET
TerminalTypeMultiTechv104Type	R	All	Config	ASSET
Terminalv100Type	R	All	Config	ASSET
Terminalv103Type	R	All	Config	ASSET
Terminalv104Type	R	All	Config	ASSET

*Note: Project104Type is CRUD provided that the Project is inactive and EDS is configured to allow Project deletion.

EDS Data Types Deprecated in this Release

This table lists the data types deprecated in this release:

EDS Type	CRUD Support	Technology	Category	Product
TerminalTypeCSType	R	All	Config	ASSET
TerminalTypeEGPRSType	R	All	Config	ASSET
TerminalTypePSType	R	All	Config	ASSET

EDS Data Types to be Deprecated at Version 10.0.5

This table lists the data types deprecated in this release:

EDS Type	EDS Type	EDS Type
AntennaDeviceType	LTENodev81Type	UMTSCarrierConfigType
AntennaPatternv801Type	LTENodev90Type	UMTSCellv81Type
CellSitev81Type	LTERepeaterv81Type	UMTSCellv90Type
CellSitev90Type	NeighbourLimitType	UMTSRepeaterv81Type
GSMCarrierType	NeighbourType	WiFiCarrierType
GSMCellv81Type	NodeBv81Type	WiFiCellType
GSMCellv90Type	RadioEquipv80Type	WiFiCellv90Type
GSMRepeaterv81Type	ServiceMultiTechType	WiFiNodeType
GSMSubCell	ServiceUMTSType	WiFiNodev90Type

EDS Type	EDS Type	EDS Type
LocationObjectHierarchyv81Type	Servicev81Type	
LocationObjectHierarchyv90Type	Servicev90Type	
LocationObjectv81Type	TerminalTypeMultiTechType	
LTECarrierType	Terminalv81Type	
LTECellv81Type	Terminalv90Type	
LTECellv90Type	TerminalTypeGSM_UMTSv81Type	

Special Changes in this Release

There are five major changes in this release.

Multi-Technology Cell

The transition from single-technology cells to the capability for multi-technology cells, which means that the cell object now has the potential to be configured for multiple technologies. This offers much greater flexibility than before, where only one technology could be configured on a cell object.

This transition began in release 9.1 with the introduction of the MU-Node, which was a new network element that can contain cells of a single technology or multiple technologies.

Given that queries against Multi-Technology Cells (and Multi-Technology Neighbours) can return a lot of data in a single round-trip for all the additional technologies a cell may offer, EDS supports an additional query parameter named “OnlyActiveTechnologyData” on each of its interfaces that will suppress supported technology parameters and only return the active.

Addition of 5G Technology

ASSET and EDS now support 5G.

For more information on the above changes, see the ‘What’s New’ section in the ENTERPRISE SUITE Help or the ENTERPRISE Release Overview and Business Impacts document. For information on the corresponding behaviour within ASSET, see the *ASSET User Reference Guide*.

For more information on configuration options for setting permissions in EDS, see Setting Permissions for Multi-Technology MU-Nodes on page 71 and Setting Permissions for Multi-Technology Cells on page 71.

JSON Data Format and OpenAPI (Swagger) RESTful R/W Interface

This release introduces an additional RESTful endpoint based upon the OpenAPI standard. This endpoint provides the same full r/w support for EDS datatypes as the SOAP interface but with the addition of serializing data as either XML or JSON based formats.

RabbitMQ Integration

Historically, EDS has used MSMQ as a means of providing increased scalability and robustness. As part of the introduction of the ASSET Event Publisher, RabbitMQ may be used as an alternative.

Special Changes in the Previous Release

There were some significant changes in the previous release (9.1). One example was the introduction of the MU-Node, which was a new network element that can contain cells of a single technology or multiple technologies. This represented the first step of the transition towards the multi-technology cell; this transition towards the multi-technology cell is now complete in EDS version 10.

If you are moving to EDS version 10 from an earlier version, it is recommended that you read the 'Special Changes in this Release' section in the EDS Technical Reference Guide 9.1, in order to be aware of the useful features that were added into EDS at that time.

Native Co-ordinate System Support

EDS natively supports the same coordinate system as ENTERPRISE through the use of the open-source GDAL library. All element types which previously showed an absolute Lon/Lat coordinate now include a secondary "stored" coordinate which contains the actual database value.

Note: You need to choose which coordinate to supply to EDS when updating locations, and in the case of the "stored" coordinate, you need to supply an appropriate EPSG code denoting which Geodetic or Project system the coordinate uses.

This fragment from a GSM cell shows how the coordinates appear:

```
<gsm80:Antenna Index="1">
      <gsm:AbsLocation EPSG="4269">
            <Longitude>-97.74209949</Longitude>
            <Latitude>30.27021592</Latitude>
      </gsm:AbsLocation>
      <gsm80:Location Relative="Relative" EPSG="0">
            <X>1.051E-05</X>
            <Y>8.92E-06</Y>
            <Z>0</Z>
      </gsm80:Location>
</gsm80:Antenna>
```

The "Location" element will contain the actual value that is stored within the database, and the "AbsLocation" element will use this stored coordinate, and, depending whether it is absolute or relative, perform the appropriate conversion to create an antenna location, including the location of the property.

When updating or creating antennas with EDS v80 types (or later), you must decide whether to specify the AbsLocation or the Location. EDS will accept both, but the Location element will take precedence.

Note: A valid EPSG code must be supplied to EDS. The list of available EDS codes can be found at http://spatialreference.org/ref/epsg/

Limitations of the EDS

The EDS does not support the following:

- ENTERPRISE Filters - the EDS employs its own query system for identifying and extracting data from the database.
- ENTERPRISE Security - all interactions with the EDS are performed on a trust basis. The basic assumption is that the EDS will be deployed as a system-to-system solution rather than as an access point. Therefore the EDS will let a user modify any supported data type which has write facility. The EDS is not intended to be made public to an internet access point, and is intended to be strictly under network security governance.
- ENTERPRISE "Diff" tables - The EDS does not interact with the multi-user Diff tables offered within ENTERPRISE. All modifications made to the database occur under a named user ID called EDS_SOA_Client, and changes are made directly to the master tables within the database.
- Database support - The EDS is written specifically against the appropriate version of the ENTERPRISE database schema, and will not work without some modification on an older (or newer) release of the product. As a result, the EDS will validate against the database version in order to ensure compliance.
- Warning message display - In ASSET, when a Logical Antenna is configured as a Switched Beam Antenna but the switch beam configuration is incomplete or incorrect, ASSET displays the value "Bad config" with a message indicating a reason. EDS is not able to display warning messages, so the message is written to the Warning log and a value of -9999 is returned.

2 About the EDS SOAP Web Service Interface

There are two service contracts available:

- Strongly Typed
- Basic

For more information, see EDS Strongly Typed Service Contract Architecture and EDS Basic Service Contract Architecture on page 49, the latter of which includes a comparison between the two architectures.

EDS Strongly Typed Web Service Interface

This section deals with the service contract and objects that make up the primary interface for manipulating data with the EDS.

The base endpoint address for this interface (WS-HTTP) is:

http://{server}:{port}/Aircom/EDS/WS

A Microsoft-compatible “netTCP” endpoint is:

netTCP://{ server}:8734/Aircom/EDS/TCP

Its WSDL can be retrieved from this URI:

http://{server}:8730/Aircom/EDS/WS?wsdl

EDS Strongly Typed Service Contract Architecture

The EDS web service uses a request/response architecture based on a simplified subset of a reference pattern created by the Liberty Alliance, a web service standards group. Specifically, it is based on the "Data Services Template" implementation.

The service contract uses a principle known as “operations as data”, and therefore the EDS only offers four basic methods for data manipulation that all accept one parameter:

- CREATE
- UPDATE
- QUERY
- DELETE

Each request contains facets which describe the data to be queried or manipulated, as well as additional metadata which controls how the service will behave in response to the request. For example, the query request will contain metadata describing the query, as well as paging parameters to indicate which part of the result set to return.

Contract Implementation

A simplified view of the service interface is provided in this picture to illustrate the operation signature of the four methods described in the above section.

All operations use XmlSerializationFormat.

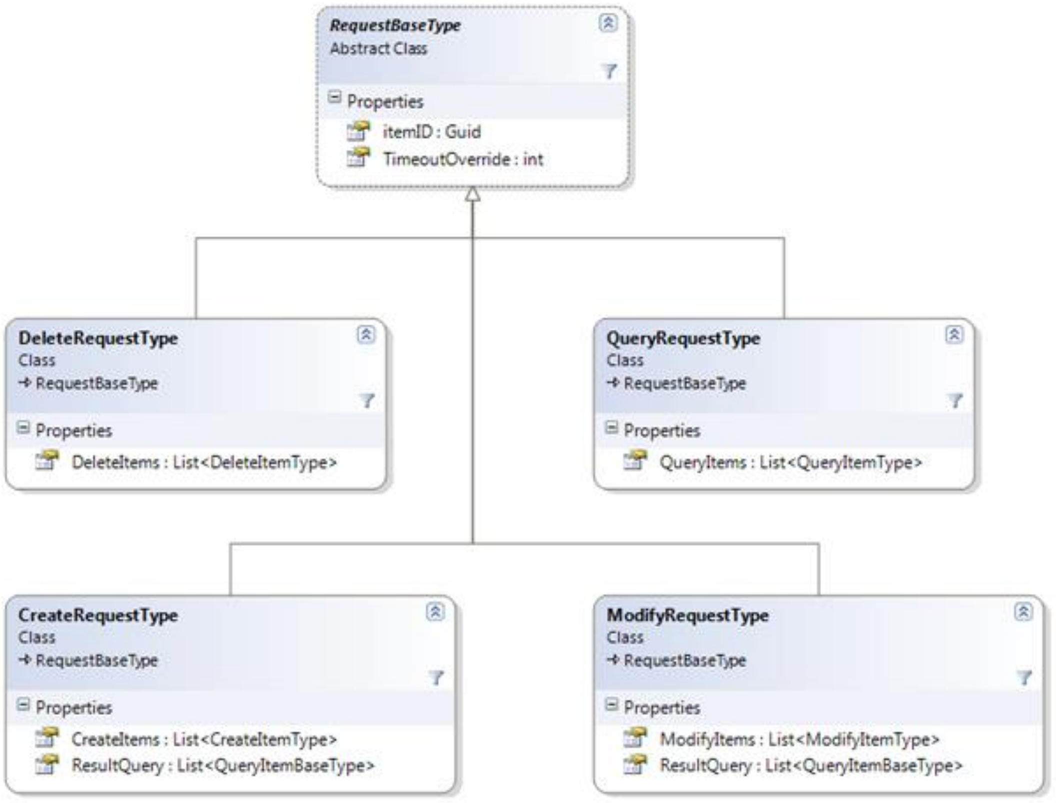
```
[ServiceContract(Namespace=NamespaceTypes.tns, Name="EDS")]
public interface IEDSService
       {
            [OperationContract]
            CreateResponseType Create(CreateRequestType data);

            [OperationContract]
            QueryResponseType Query(QueryRequestType data);

            [OperationContract]
            ModifyResponseType Modify(ModifyRequestType data);

            [OperationContract]
            DeleteResponseType Delete(DeleteRequestType data);

            [OperationContract]
            SupportedTypes QueryableTypes();

            [OperationContract]
            SupportedTypes WriteableTypes();

            [OperationContract]
            SystemStatus Status();

            [OperationContract]
            HistoricalSystemStatus HistoricalStatus();

       }
```

The four ancillary methods on the service contract perform the following:

- QueryableTypes - returns an enumeration indicating all available data types within the EDS
- WriteableTypes - a subset of the previous enumeration
- Status & HistoricalStatus - used for performance monitoring

Each request operation shares a common request base type, and elements that are specific to that operation.

This picture shows the hierarchy for these four request types:



Request Type Hierarchy

Each request type has a unique ItemID used to correlate the message through the system, a collection of sub-items which allow for batching of multiple data types within a single request, and, depending on the operation, a ResultQuery which allows for the inclusion of a query or queries after the data manipulation has completed.

This table describes the common properties contained in the picture:

Property Name	Type	Description
itemID	GUID	A unique identifier used to correlate the request through the system between the service internal layers and the client.
TimeoutOverride	Int	The EDS is configured to timeout any request which takes longer than 45 seconds to complete. This can occur when the server is busy, or the request is complex and therefore requires more time to process. By supplying a timeout value, the presentation service will wait for the new interval, or until it receives a response from the server. **Note:** Client developers should be aware of this value, and choose a suitable override if you anticipate that the request will take longer than the default value.
Runsequentially	bool	By default EDS will run each request item in a separate thread (up to 8). Setting this to true will instruct EDS to run multiple request items in the same thread.
DeleteItems	List<DeleteItemType>	A collection of request items which allow for multiple objects or object types to be deleted within a single request.
ModifyItems	List<ModifyItemType>	A collection of request items which allow for multiple types of objects to be updated within a single request.
ResultQuery	List<QueryItemBaseType>	Once a modification or create request has completed, you might want to request a dataset in order to save round trips.

Request Type Schemas

This section contains schema extracts for the types discussed under Contract Implementation on page 49.

All Requests derive from the same base Request EWSRequestBaseType.

The schema extract for EWSRequestBaseType is:

```
<xs:complexType name="EWSRequestBaseType" abstract="true">
      <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="1" ref="q1:itemID"
      xmlns:q1="http://www.aircominternational.com/contract/Util/2009/10"/>
            <xs:element minOccurs="0" maxOccurs="1" name="masterID" type="q2:guid"
            xmlns:q2="http://microsoft.com/wsdl/types/"/>
      </xs:sequence>
</xs:complexType>
```

RequestBase Type

The schema extract for RequestBaseType is:

```
<xs:complexType name="RequestBaseType" abstract="true">
      <xs:complexContent mixed="false">
            <xs:extension base="q1:EWSRequestBaseType"
xmlns:q1="http://www.aircominternational.com/contract/EWS/2011/01">
                  <xs:sequence>
                        <xs:element minOccurs="1" maxOccurs="1"
                         name="TimeoutOverride" type="xs:int"/>
                        <xs:element minOccurs="0" maxOccurs="1"
                         name="RunSequentially" type="xs:boolean"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

CreateRequestType

The schema extract for CreateRequestType is:

```
<xs:complexType name="CreateRequestType">
      <xs:complexContent mixed="false">
            <xs:extension base="q1:RequestBaseType"
             xmlns:q1="http://www.aircominternational.com/contract/EDS/
             DST/2009/10">
                  <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="unbounded"
             name="CreateItems" type="tns:CreateItemType"/>
                        <xs:element minOccurs="0" maxOccurs="unbounded"
             name="ResultQuery" type="tns:QueryItemBaseType"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

ModifyRequestType

The schema extract for ModifyTypeRequest is:

```
<xs:complexType name="ModifyRequestType">
      <xs:complexContent mixed="false">
            <xs:extension base="q2:RequestBaseType"
xmlns:q2="http://www.aircominternational.com/contract/EDS/DST/2009/10">
                  <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="unbounded"
                         name="ModifyItems"
                        type="tns:ModifyItemType"/>
                        <xs:element minOccurs="0" maxOccurs="unbounded"
                         name="ResultQuery"
                        type="tns:QueryItemBaseType"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

DeleteRequestType

The schema extract for DeleteRequestType is:

```
<xs:complexType name="DeleteRequestType">
      <xs:complexContent mixed="false">
            <xs:extension base="q15:RequestBaseType"
xmlns:q15="http://www.aircominternational.com/contract/EDS/DST/2009/10">
                  <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="unbounded"
                         name="DeleteItems"
                        type="tns:DeleteItemType"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

QueryRequestType

The schema extract for QueryRequestType is:

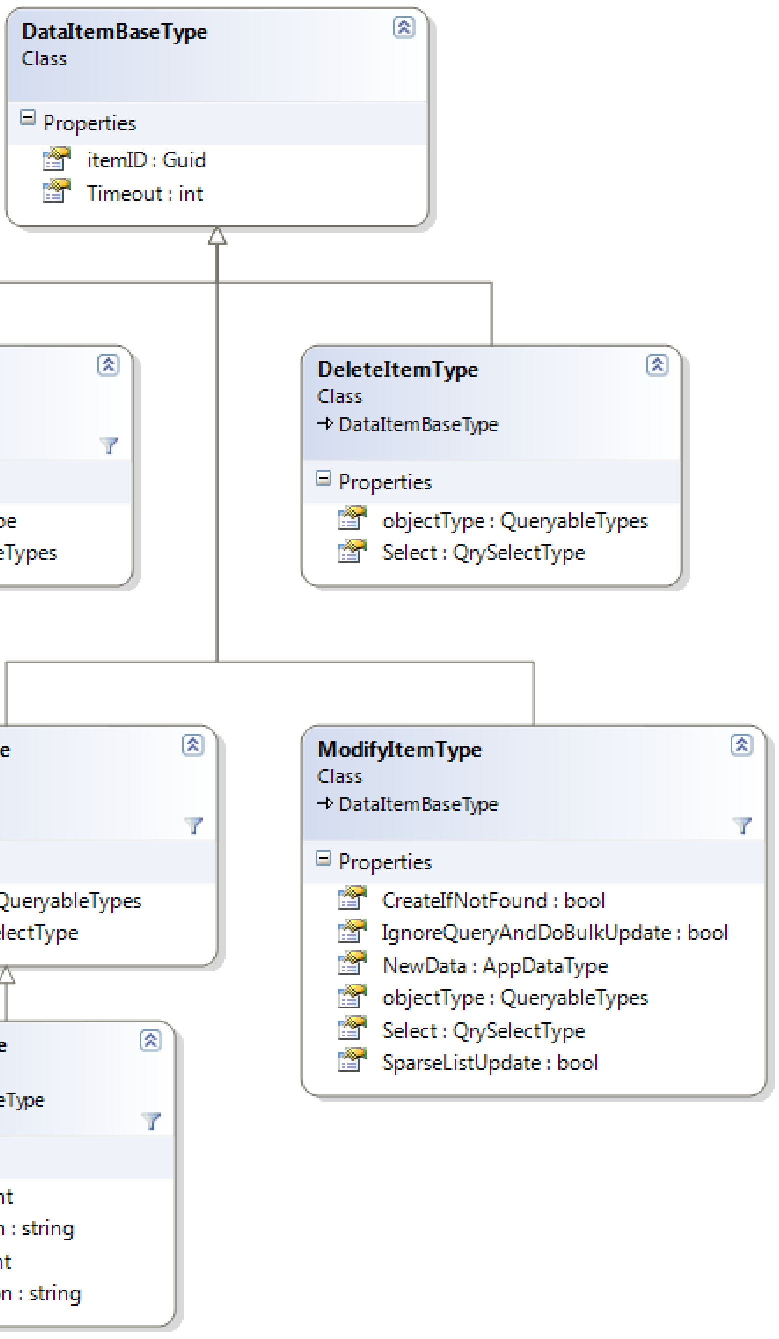
```
<xs:complexType name="QueryRequestType">
      <xs:complexContent mixed="false">
            <xs:extension base="q16:RequestBaseType"
xmlns:q16="http://www.aircominternational.com/contract/EDS/DST/2009/10">
                  <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="unbounded"
                         name="QueryItems"
                        type="tns:QueryItemType"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

Request Items

Each request type (discussed under Contract Implementation on page 49) decomposes into a list of request items. These request items contain the main detail of the request, and each request item can only be applicable for a single supported object type. This means that if you want to include a batch request that may optionally want to manipulate several object types within a single round trip, you would need to use multiple request items.

Note: The EDS will prohibit requests which ask for more than 500 rows of data within a single request, therefore multiple request items may be used to return larger row sets through the use of pagination.

This picture illustrates the structure of the request items:



The hierarchy of request items follows a similar pattern to that of their parent containing types. Each request item contains its own common unique itemID to correlate the sub-request through the system. However, their contract signature varies from that of the request in order to be more specific to the operation they are representing. This table describes the properties available to EDS operations that will influence their behaviour:

Property Name	Type	Description
itemID	GUID	A unique identifier used to correlate the request through the system between the service internal layers and the client.
NewData	AppDataType	A collection whose type is defined by the objectType parameter which represents the actual data being created or modified by the request. Within this collection would be the actual telecoms specific data, such as NodeBs, Sectors and so on.
objectType	enum	An enumeration of predefined values which indicate the supported object types by the system.
Select	QrySelectType	A query indicating which objects to retrieve from the system by allowing for parameter sets and simple predicate logic.
CreateIfNotFound	bool	A flag indicating that if a modification query (Select) does not retrieve a value, the system will attempt to create the item instead.
count	int	Indicates that the query will return no more than a given number of rows, which is useful for pagination requests. Note: The maximum number of rows is 500.
offset	int	Allows you to request data from a different position within a rowset, which is useful for pagination requests.
IgnoreQueryAndDo BulkUpdate	bool	This parameter is used to instruct EDS to perform a batch update when supplied a block of data rather than use the query provided to locate the record to be updated. Setting the parameter to true forces EDS to use the IID and the BVID of the supplied object to find a match. In other words, rather than requiring an EDS selection query to locate data to update, EDS will implicitly use the dataset it has been handed to locate the corresponding item within the database.
exclusion	string	A pipe ("|") delimited string specifying sections of the EDS datatype to ignore. This can be used to create more efficient fetches from the service. For more information, see Query Result Inclusion and Exclusion.
inclusion	string	A pipe ("|") delimited string specifying sections of the EDS datatype to include. This can be used to create more efficient fetches from the service. For more information, see Query Result Inclusion and Exclusion.
projection	string	Allows EDS to transform the result of one type into a custom output. Currently this feature is not available.
SparseListUpdate	bool	Instructs EDS to preserve existing list entries if not included during an UpdateRequest. Default behaviour is to replace.
FileItem	FileItem	Instructs EDS to look for a URI location to either read from or write to depending on the request. If a URI is supplied then an optional Impersonation item is available to pass user credentials via as Impersonation property
Impersonation	Impersonation	Allows EDS to access a URI using these credentials. To access the URI, a Domain name, User name and Password must be supplied that has access to that URI
CreateNewAntennas	bool	For 9.1 Cell Types, the Logical Antenna is exposed on the Feeders within the Cells. In previous versions you could set Logical Antenna Index to ‘-1’ and the system would generate the next unique index. In 9.1, this is still possible, but only if you pass in ‘CreateNewAntennas’ set to ‘true’. (For Create and Update requests.)

Property Name	Type	Description
ForceAntennaCreation OnConflict	bool	In previous versions, if you tried to add a new Logical Antenna via a legacy (pre-9.1) Node/CellSite, and that new LogicalAntenna Index already exists, then the update will fail, which prevents the original Logical Antenna becoming accidentally overwritten. In 9.1, if you do not know the correct new index you can pass in 'ForceAntennaCreationOnConflict' set to 'true', so that if the index already exists, the system will generate the next unique index and add the Logical Antenna. (This is for Create and Update requests only)
AddUnassigned Carrier	bool	In previous versions, when an LTE, UMTS and Wi-Fi Cell type is added or updated and the Carrier that is selected in the Cell has not been assigned to the Parent Node, then EDS will reject the transaction. In 9.1, you can set 'AddUnassignedCarrier' to 'true', so that if the Carrier specified is a valid Carrier, the Carrier will be allocated to the Parent Node and the transaction will succeed. (This is for Create and Update requests only.)
ImpersonateUser	string	Allows EDS to act as an 'ASSET Logged-in User'. This relates to Update, Create and Delete requests. In the Persistence config file you can now turn 'on' and 'off' the ability to impersonate an ASSET User and set the Default Impersonator User / Impersonation Group that you want the EDS Service to Impersonate, this user must exist as a genuine ASSET User and be a member of the supplied Group. If the supplied Impersonation Group does not exist or the supplied Impersonate User is not in the Supplied Impersonation Group, an error message is issued and the service shuts down. The default state for Impersonate User is 'off'. When Impersonate User is turned 'on' you can also impersonate a user at the time of Request, that is, for each Update, Create and Delete request you send to EDS, you can also send a user name and this will take precedence over the Impersonate User from the config file. Again, the user must be a valid ASSET User and a member of the Impersonation Group defined in the config file.
OnlyActiveTechnology Data	bool	For 10.0 Cell types, 10.0 Feeder types and 10.0 Neighbour types, the request will return all configured technology data. If you want to be more specific, this parameter (if set to 'true') will only return the active technology (for a given cell) instead. The default value is 'false'.
IgnoreQueryAndDo BulkDelete	bool	Similar to the IgnoreQueryAndDoBulkUpdate parameter, this setting will use a supplied block of data passed to EDS to generate a delete query against the database rather than requiring the user to craft individual "DeleteItem" requests containing a singular query. This will lead to a more efficient way of deleting multiple objects from the ASSET database as part of a single transaction.

Request Item Schemas

DataItemBaseType

The schema extract for DataItemBaseType is:

```
<xs:complexType name="DataItemBaseType">
      <xs:sequence>
            <xs:element minOccurs="1" maxOccurs="1" ref="q5:itemID"

xmlns:q5="http://www.aircominternational.com/contract/Util/2009/10"/>
      </xs:sequence>
</xs:complexType>
```

CreateItemType

The schema extract for CreateItemType is:

```
<xs:complexType name="CreateItemType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:DataItemBaseType">
                  <xs:sequence>
                        <xs:attribute xmlns:q9="…" ref="q9:objectType" />
                        <xs:attribute xmlns:q10="…"
ref="q10:UseQueryCache" />
                        <xs:attribute
name="SuppressLogicaltoPhysicalAntennaCorrection"
                         type="xs:boolean" use="required" />
                        <xs:attribute name="SuppressAntennaUpdatesAtCell"
                         type="xs:boolean" use="required" />
                        <xs:attribute
name="ForceAntennaCreationOnConflict"
                         type="xs:boolean" use="required" />
                        <xs:attribute name="CreateNewAntennas"
                         type="xs:boolean" use="required" />
                        <xs:attribute name="AddUnassignedCarrier"
                         type="xs:boolean" use="required" />
                        <xs:attribute name="ImpersonateUser"
type="xs:string" />
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
```

DeleteItemType

The schema extract for DeleteItemType is:

```
<xs:complexType name="DeleteItemType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:DataItemBaseType">
                  <xs:sequence>
                        <xs:element
xmlns:q15="http://www.aircominternational.com/
                         Schemas/Query/2009/09" minOccurs="1"
maxOccurs="1"
                         name="objectType" type="q15:QueryableTypes" />
                        <xs:element
xmlns:q16="http://www.aircominternational.com/
                        Schemas/Query/2009/09" minOccurs="0"
maxOccurs="1"
```

```
                         name="Select" type="q16:QrySelectType" />
                        <xs:element minOccurs="0" maxOccurs="1"
                         name="ExistingData" type="tns:AppDataType" />
                  </xs:sequence>
                  <xs:attribute name="ImpersonateUser" type="xs:string"
/>
                  <xs:attribute name="IgnoreQueryAndDoBulkDelete"
type="xs:boolean" use="required" />
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

QueryItemBaseType

The schema extract for QueryItemBaseType is:

```
<xs:complexType name="QueryItemBaseType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:DataItemBaseType">
                  <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="1"
                         name="Select" type="q13:QrySelectType"
                        xmlns:q13="http://www.aircominternational.com/
                         Schemas/Query/2009/09"/>
                  </xs:sequence>
                  <xs:attribute  name="objectType"
type="q14:QueryableTypes"
                   use="required"
xmlns:q14="http://www.aircominternational.com/
                   Schemas/Query/2009/09"/>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

QueryItemType

The schema extract for QueryItemType is:

```
<xs:complexType name="QueryItemType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:QueryItemBaseType">
                  <xs:sequence>
                        <xs:attribute name="count" type="xs:int"
use="required" />
                        <xs:attribute name="offset" type="xs:int"
use="required" />
                        <xs:attribute name="projection" type="xs:string"
/>
                        <xs:attribute name="exclusion" type="xs:string"
/>
                        <xs:attribute name="inclusion" type="xs:string"
/>
                        <xs:attribute name="onlyActiveTechnologyData"
type="xs:boolean" use="required" />
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
</xs:complexType>
```

ModifyItemType

The schema extract for ModifyItemType is:

```
<xs:complexType name="ModifyItemType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:DataItemBaseType">
                  <xs:sequence>
                        <xs:element xmlns:q13="…" minOccurs="0"
maxOccurs="1"
                         name="Select" type="q13:QrySelectType" />
                        <xs:element minOccurs="0" maxOccurs="1"
                         name="NewData" type="tns:AppDataType" />
                  </xs:sequence>
                  <xs:attribute xmlns:q14="…"  name="objectType"
                   type="q14:QueryableTypes" use="required" />
                  <xs:attribute name="CreateIfNotFound"
                   type="xs:boolean" use="required" />
                  <xs:attribute name="IgnoreQueryAndDoBulkUpdate"
                   type="xs:boolean" use="required" />
                  <xs:attribute name="SparseListUpdate"
                   type="xs:boolean" use="required" />
                  <xs:attribute
name="SuppressLogicaltoPhysicalAntennaCorrection"
                   type="xs:boolean" use="required" />
                  <xs:attribute name="SuppressAntennaUpdatesAtCell"
                   type="xs:boolean" use="required" />
                  <xs:attribute name="ForceAntennaCreationOnConflict"
                   type="xs:boolean" use="required" />
                  <xs:attribute name="CreateNewAntennas"
                   type="xs:boolean" use="required" />
                  <xs:attribute name="AddUnassignedCarrier"
                   type="xs:boolean" use="required" />
                  <xs:attribute name="ImpersonateUser"
                   type="xs:string" />
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

Application Specific Data Types

A request may include an AppDataType, where applicable. This type contains the actual collection of data which is supported by the system. The structure of the AppDataType class contains a collection of RootEntityType, which represents the base type for all network data supported by the system.

The RootEntityType describes the minimum amount of data which all supported elements must contain:

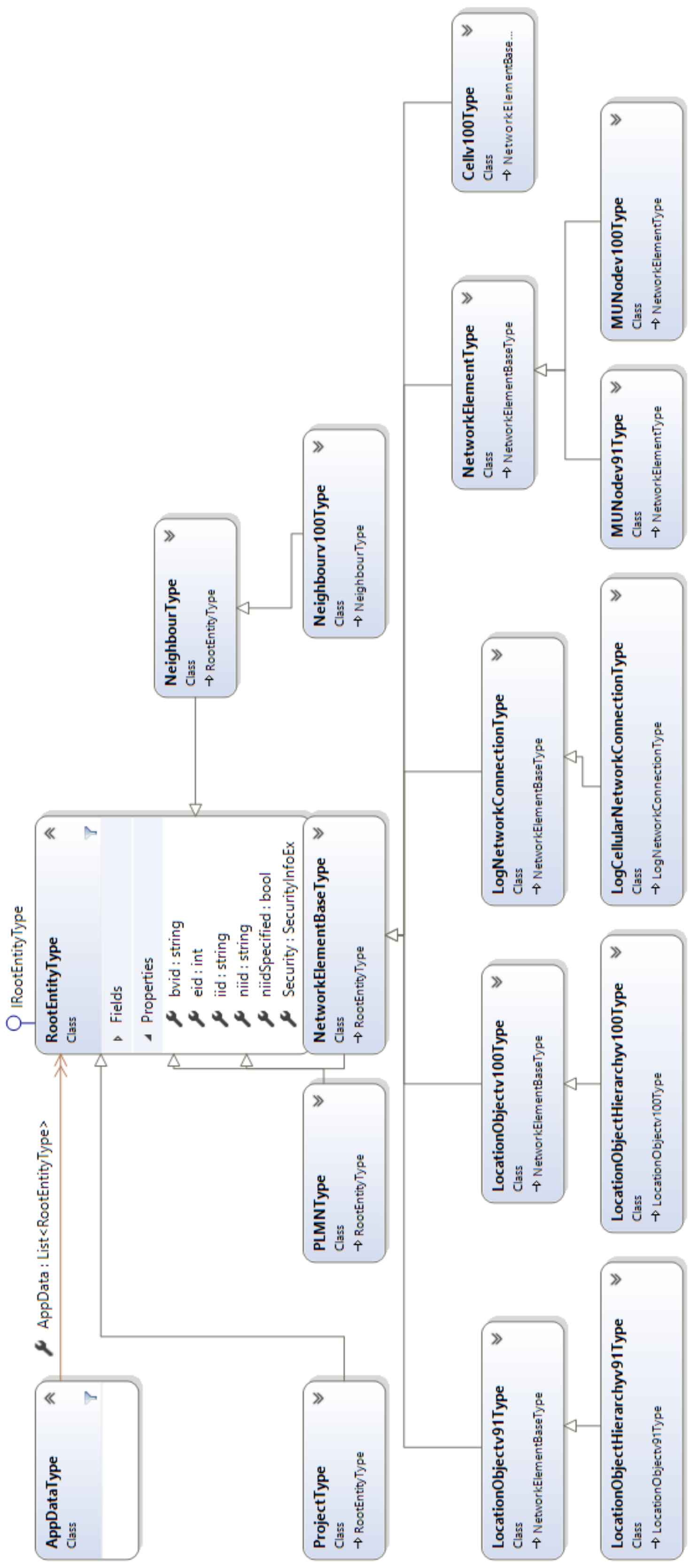
- iid – Unique name of the element
- bvid – Owning project of the element

The following are optional or only necessary when renaming elements (in the case of niid)

- eid – Internal DB key of the element
- niid – Used to rename the identity of a given data-type (where supported)

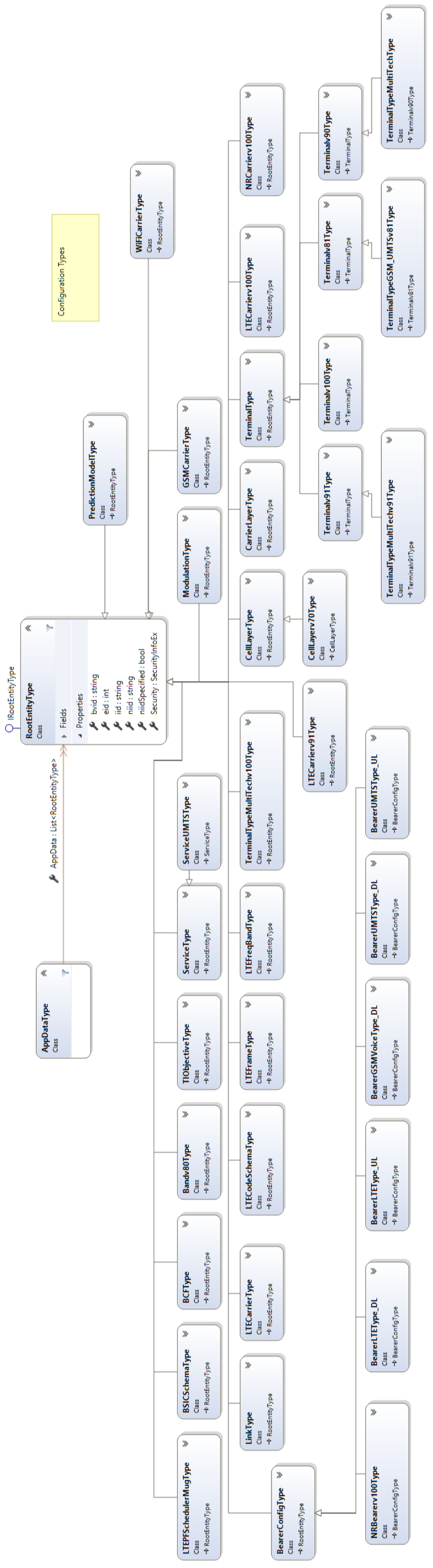
The following diagrams show the supported types from EDS.

Common Types

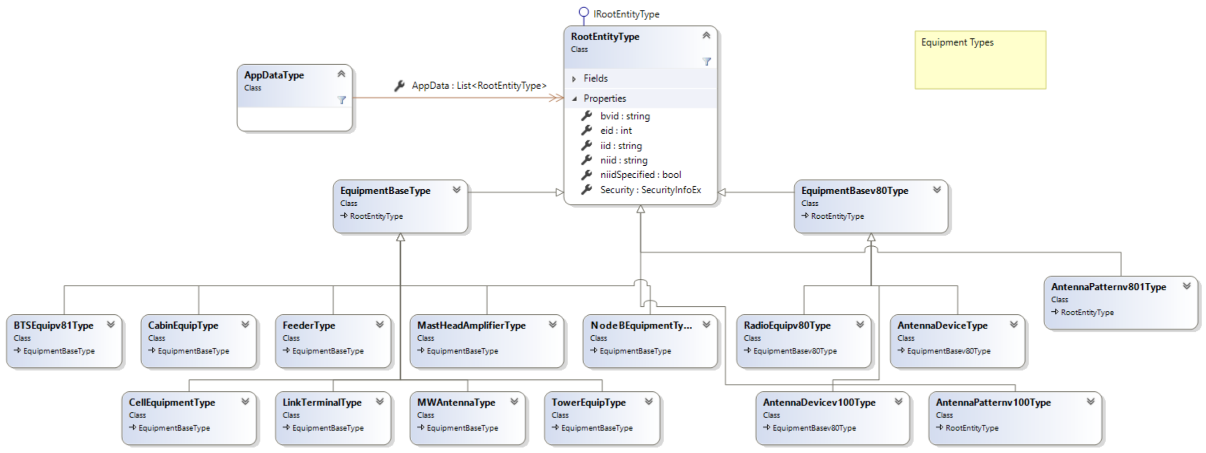


Note: For brevity this diagram does not include earlier versions of supported data types (such as v90, v81).

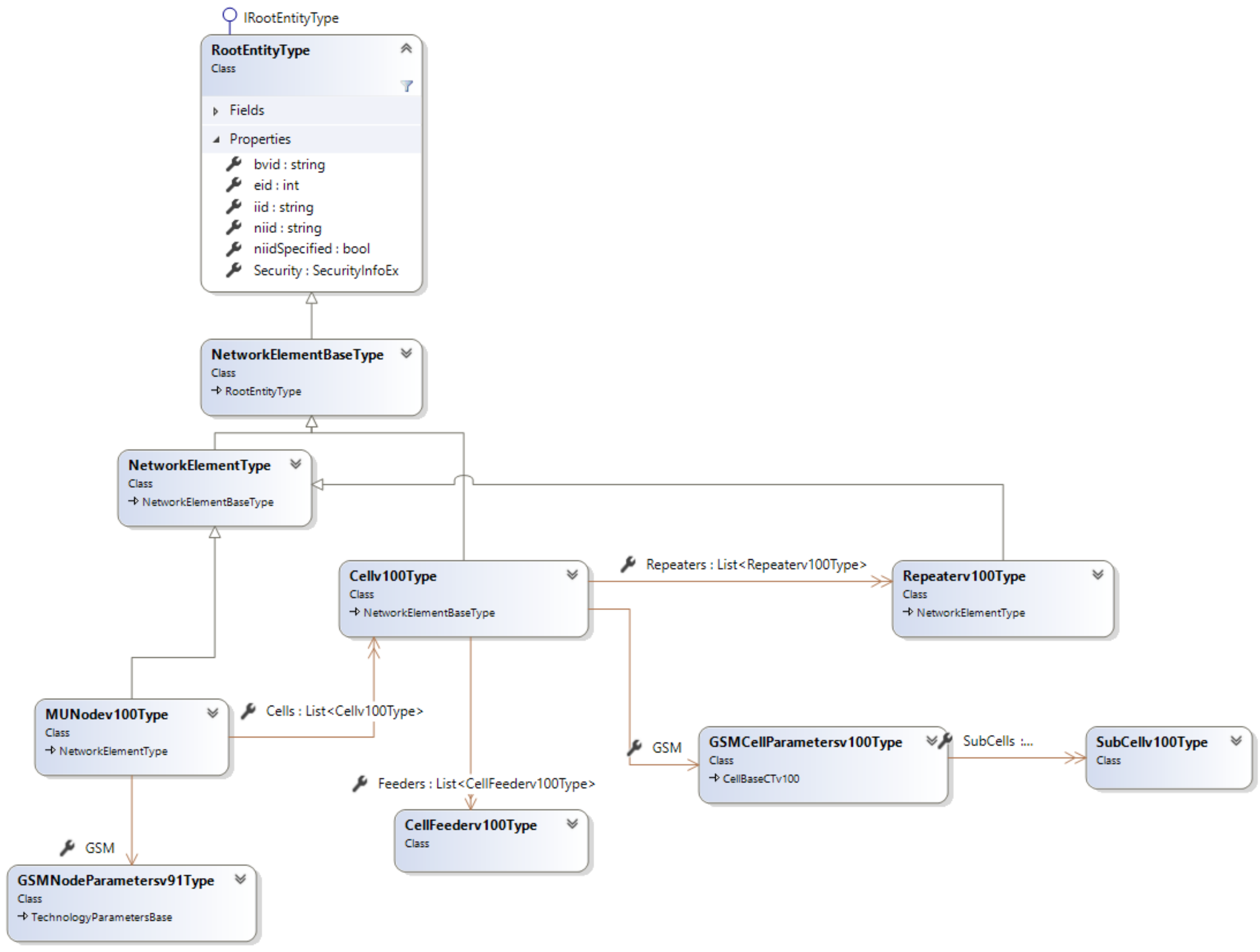
Configuration Types



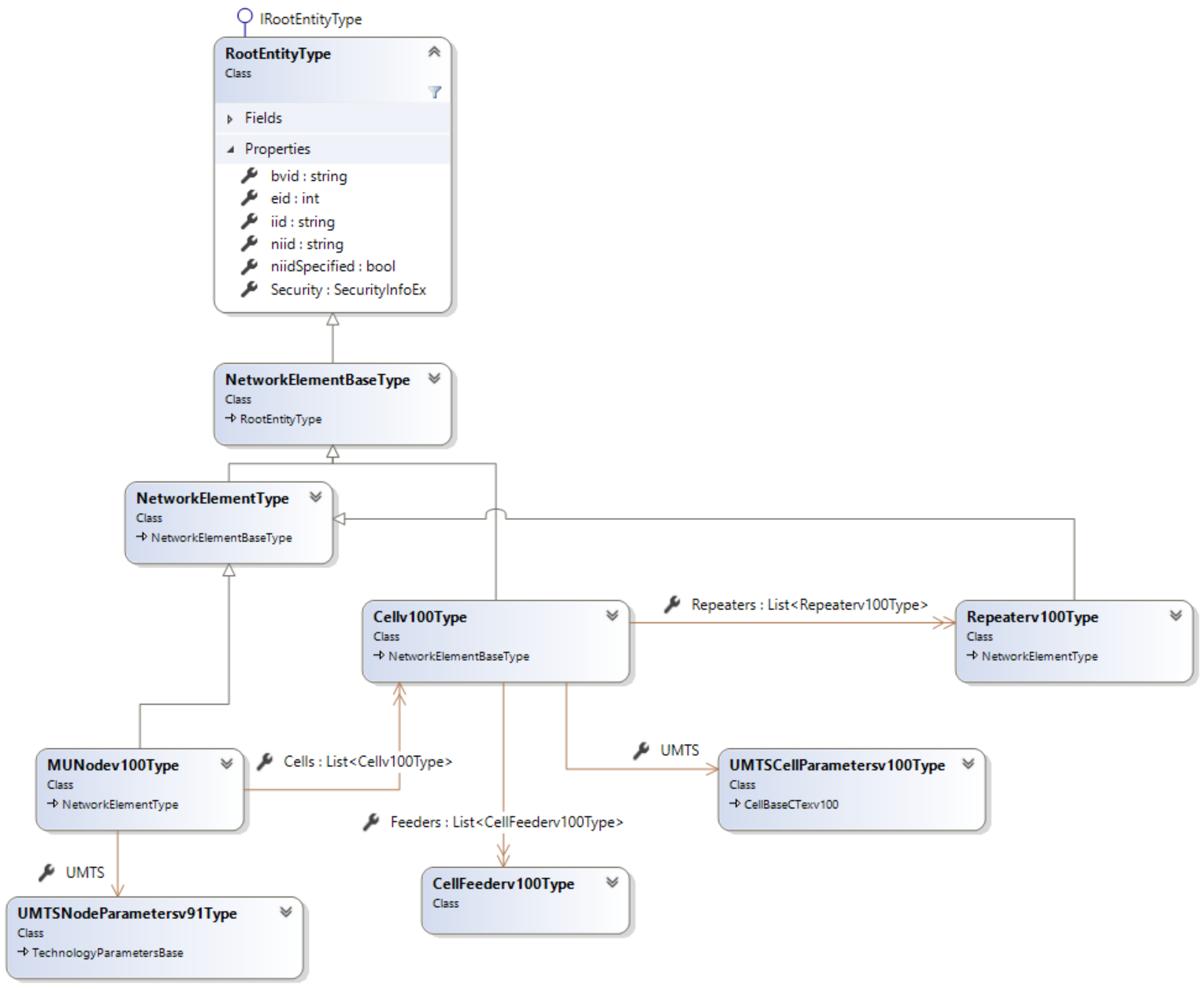
Equipment Types



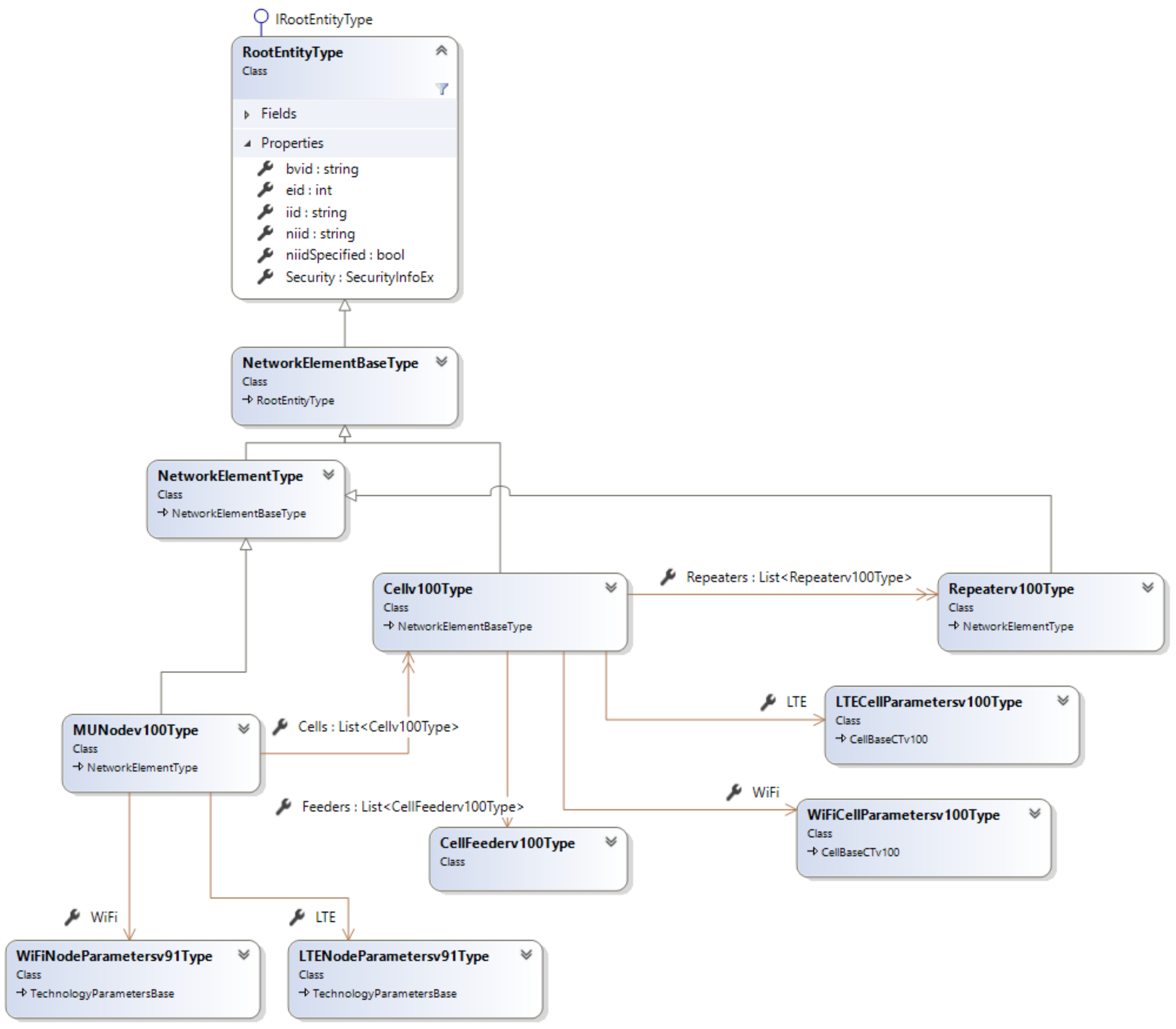
Simplified v10.0 GSM Types



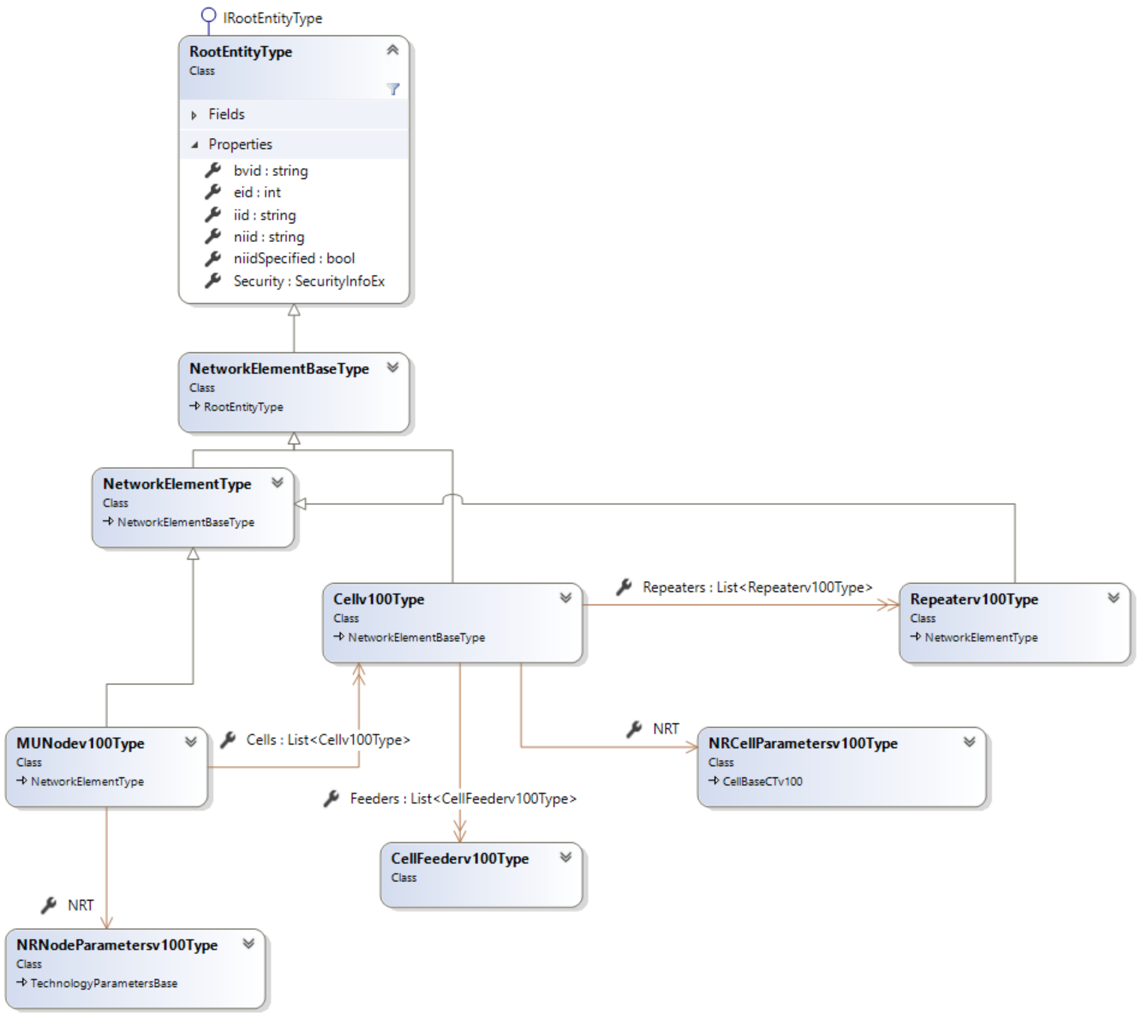
Simplified v10.0 UMTS Types



Simplified v10.0 LTE and Wi-Fi Types

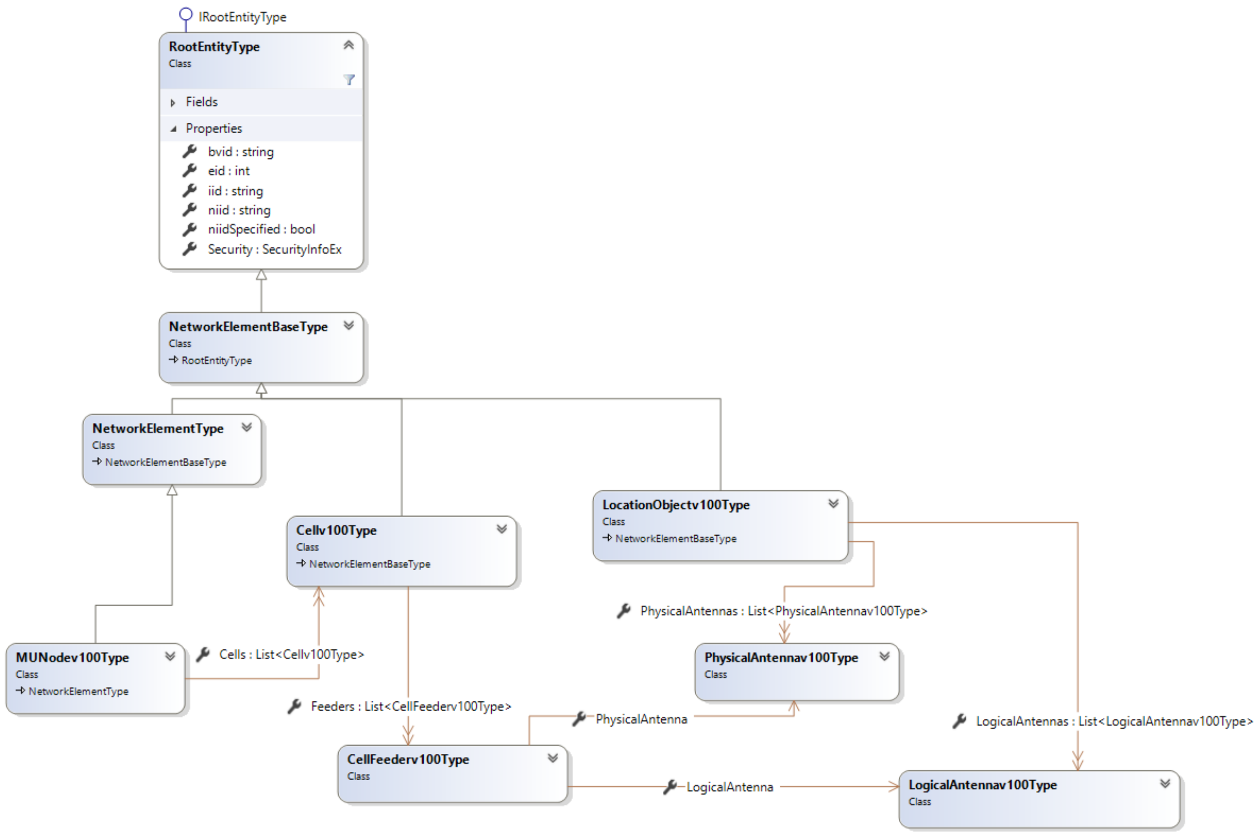


Simplified v10.0 5G NR Types



Cellular Antenna Connectivity in v10.0

This diagram illustrates the relationship between a v10 ASSET Property type (LocationObject), a Multi-Tech Node (MUNode) and an Multi-Tech Cell (Cell). Although not shown in the diagram, as well as the reference to the LocationObject via the Feeder, both the MUNode and Cell will have a unique reference of their own given LocationObject. This is to allow configurations where the Node, Cell and the Feeder may reference equipment located on multiple physical locations.



AppDataType and RootEntity Type Schemas

This section contains the XML schema fragments for the AppDataType and the RootEntityType.

Note: For brevity, none of the derived types from RootEntityType are included.

AppDataType

The schema extract for AppDataType is:

```
<xs:complexType name="AppDataType">
      <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded"
name="AppData"
            type="q9:RootEntityType"
xmlns:q9="http://www.aircominternational.com/Schemas/CommonTypes/2009/05"
/>
      </xs:sequence>
</xs:complexType>
```

RootEntityType

The schema extract for RootEntityType is:

```
<xs:complexType name="RootEntityType">
      <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="1" form="unqualified"
             name="Security" type="tns:SecurityInfo"/>
      </xs:sequence>
      <xs:attribute  name="iid" type="xs:string"/>
      <xs:attribute  name="bvid" type="xs:string"/>
      <xs:attribute  name="eid" type="xs:int"/>
</xs:complexType>
```

Query Result Caching

By default, EDS will cache certain shared equipment or configuration types (for example, antennas or projects) to yield better performance when fetching data. This may occasionally interfere when new data has been added externally to EDS, for example through ENTERPRISE or directly into the database. If the expected data does not appear in EDS, you can try disabling the cache temporarily for the query you are supplying. This can be done by setting the “OverrideCacheDefault” element to false in the via the UseQueryCache attribute.

Important: It is recommended to use this feature if you are bulk-fetching many rows of data from EDS.

Query Result Inclusion and Exclusion

By default, when returning a result for a given type, EDS will assemble the full element as defined by its XML schema definition – assuming the underlying data exists. It is possible to instruct EDS to either include or exclude certain segments of an element’s XML hierarchy by using these features. This can significantly improve EDS performance, as well as reducing data volume, especially if the client only requires a sparse object.

The following notes apply to both include and exclude functionality:

'Include'

- By specifying the keyword ‘Include’ on the service contract, EDS only selects the elements you specify, or the prerequisite path to the elements if contained within a sub component such as Cells or Antennas for example.
- Any ‘Include’ instruction will take priority over an ‘Exclude’ instruction if both appear in the same request.
- Irrespective of the path(s) specified in an ‘Include’, EDS will always attempt to emit “iid” and “bvid” attributes in the output data although it is possible to create “unusual” EDS output configurations that may be rejected upon subsequent input of the same data back into EDS.
- If an ‘Include’ fails to match any appropriate elements, EDS will output a minimal type containing just the iid and bvid.

'Exclude'

- ‘Exclude’ behaves in the opposite way to ‘Include’, because it only subtracts elements where a matching path exists and thereby preserving the elements around it.
- Most elements support both ‘Exclude’ and ‘Include’ whereas in the previous EDS releases only “sub-list” groups such as Cells, Antennas or CustomFields were permitted.
- It is possible to create invalid EDS configurations using ‘Exclude’ that may be rejected due to validation failure if the same structure is used for subsequently performing an EDS ‘Create’ or ‘Update’.

Syntax

To use either feature, the client may simply name the element(s) they wish to include or exclude and EDS will attempt to match them while assembling the result following a query. Multiple inclusions or exclusions may be specified using the pipe "|" delimiter; sub-list exclusions are referenced using the "/".

The feature supports limited wildcards being present in the pattern as well as regular expressions:

- To use a wildcard, the client may include an asterisk "*" as part of the pattern for example "/Cells/*". If the asterisk is omitted then the rule would not necessarily traverse to the descendent parameters beneath each cell.
- To specify a regular expression, the pattern must include angular braces around the expression itself "{expr}". It is still possible to separate multiple patterns using the pipe "|" delimiter.

Note: Regarding the usage of regular expressions, they are expressed literally against the path being evaluated, not hierarchically (for instance where ‘Include’ will select ancestors on a path such as /cells/feeders). This means that if a regular expression were "{a.*}" for example, it would select all elements starting with "A" (case insensitive) at any level where it is permitted to traverse.

For best results with usage of this feature it is worth combining them with the more traditional syntax for ease of use, for example "/cells/feeders/*|{a.*}"

This table shows some example exclusions:

This instruction:	Will give this result:
Security	Skips construction of the common Security element.
Security|CustomFields	Skips construction of the common Security and User Defined Fields element.
Security|Cells/Security|Cells/ SubCells	Skips construction of the common Security element on a CellSite and the security element on its attached Cells as well as the SubCell.
Cells/Carrier|Cells/ Resources|Cells/ HSDPA|Cells/HSUPA	Skips construction of the Carriers on attached UMTS Cells within a NodeB, the Cell resources and the Cell HSPA parameters.
{Cells/c.*}	Skips construction of all elements beneath the Cell beginning with "c" using a regular expression
Cells	Excludes parent element Cells
Cells/*	Includes parent element Cells but truncates list of Cells to iid/bvid.

This table shows some example inclusions:

This instruction:	Will give this result:
Security/*	Includes only Security element and its descendents.
Security/*|CustomFields/*	Includes only the common Security and User Defined Fields elements.
Cells/Carrier/*|Cells/ Resources/*|Cells/HS	Includes only the Carriers on attached UMTS Cells within a NodeB, the Cell resources and the Cell HSPA and HSUPA parameters.
{security}|Cells	Matches only the Security element on both the root and cell level.

Important: The ‘Query Result Inclusion and Exclusion’ feature is available to all interfaces supported by EDS (SOAP, REST, Loader). However, it is strongly recommended that you experiment with the syntax using the REST interface for easiest visualisation of the result.

Query Type Schemas

QryBaseType

The schema extract for QryBaseType is:

```
<xs:complexType name="QryBaseType">
      <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="1"
             name="OverrideCacheDefault">
                  <xs:complexType>
                        <xs:attribute default="false"
                         name="UseQueryCache" type="xs:boolean"/>
                  </xs:complexType>
            </xs:element>
      </xs:sequence>
</xs:complexType>
```

QrySelectType

The schema extract for QrySelectType is:

```
<xs:complexType name="QrySelectType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:QryBaseType">
                  <xs:sequence>
                        <xs:choice minOccurs="1" maxOccurs="1">
                              <xs:element minOccurs="0" maxOccurs="1"
                               name="Type"
                              type="tns:QryListElementType"/>
                              <xs:element minOccurs="0" maxOccurs="1"
                               name="Not"
                              type="tns:QryNotPredicateType"/>
                              <xs:element minOccurs="0" maxOccurs="1"
                               name="And"
                              type="tns:QryAndPredicateType"/>
                              <xs:element minOccurs="0" maxOccurs="1"
                               name="Attribute"
                              type="tns:QryAttributeBaseType"/>
                              <xs:element minOccurs="0" maxOccurs="1"
                               name="Extension"
                               type="tns:QryExtensionPredicateType"/>
                              <xs:element minOccurs="0" maxOccurs="1"
                               name="Simple"
type="tns:QrySimpleTextType"/>
                              <xs:element minOccurs="0" maxOccurs="1"
                               name="Or" type="tns:QryOrPredicateType"/>
                        </xs:choice>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

QrySimpleTextType

The schema extract for QrySimpleTextType is:

```
<xs:complexType name="QrySimpleTextType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:QryBaseType">
                  <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="1"
                         name="Query" type="xs:string"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

QryAttributeBaseType

The schema extract for QryAttributeBaseType is:

```
<xs:complexType name="QryAttributeBaseType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:QryBaseType">
                  <xs:attribute name="op" type="tns:
                   QryOperationType" use="required"/>
                  <xs:attribute name="value" type="xs:string"/>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

QryUserDefinedFieldType

The schema extract for QryUserDefinedFieldType is:

```
<xs:complexType name="QryUserDefinedFieldType">
      <xs:complexContent mixed="false">
            <xs:extension base="q1:QryBaseType" xmlns:

q1="http://www.aircominternational.com/Schemas/Query/2009/09">
                  <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="1"
                         form="unqualified" name="CustomFieldGroup"
                        type="xs:string"/>
                        <xs:element minOccurs="1" maxOccurs="1"
                         form="unqualified" name="CustomFieldType"
                        type="tns:QryUserDefinedFieldGroupType"/>
                  </xs:sequence>
                  <xs:attribute name="name" type="tns:
                   QryUserDefinedFieldAttributes" use="required"/>
                  <xs:attribute name="op" type="q1:
                   QryOperationType" use="required"/>
                  <xs:attribute name="value" type="xs:string"/>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

QryAndPredicateType

The schema extract for QryAndPredicateType is:

```
<xs:complexType name="QryAndPredicateType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:QryBaseType">
                  <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="1"
                         name="lhs" type="tns:QrySelectType"/>
                        <xs:element minOccurs="0" maxOccurs="1"
                         name="rhs" type="tns:QrySelectType"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

QryOrPredicateType

The schema extract for QryOrPredicateType is:

```
<xs:complexType name="QryOrPredicateType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:QryBaseType">
                  <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="1"
                         name="lhs" type="tns:QrySelectType"/>
                        <xs:element minOccurs="0" maxOccurs="1"
                         name="rhs" type="tns:QrySelectType"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

QrySimpleListType

The schema extract for QrySimpleListType is:

```
<xs:complexType name="QrySimpleListType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:QryBaseType">
                  <xs:sequence>
                        <xs:choice minOccurs="0" maxOccurs="unbounded">
                              <xs:element minOccurs="0" maxOccurs="1"
                               name="Item" type="tns:ListItemType"/>
                              <xs:element minOccurs="0" maxOccurs="1"
                               name="NetworkItem"
type="tns:ListNetworkItemType"/>
                              <xs:element minOccurs="0" maxOccurs="1"
                               name="NbrItem"
type="tns:ListNbrItemType"/>
                        </xs:choice>
                  </xs:sequence>
            </xs:extension>
      /xs:complexContent>
</xs:complexType>
```

The schema extract for ListItemType is:

```
<xs:complexType name="ListItemType">
      <xs:attribute ref="q1:iid"
xmlns:q1="http://www.aircominternational.com/Schemas/CommonTypes/2009/05"
/>
      <xs:attribute ref="q2:bvid"
xmlns:q2="http://www.aircominternational.com/Schemas/CommonTypes/2009/05"
/>
      <xs:attribute ref="q3:eid"
xmlns:q3="http://www.aircominternational.com/Schemas/CommonTypes/2009/05"
/>
</xs:complexType>
```

The schema extract for ListNetworkItemType is:

```
<xs:complexType name="ListNetworkItemType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:ListItemType">
                  <xs:attribute name="NetworkDiscriminator" type="q4:
                   NetworkDiscriminatorType"
                   xmlns:q4="http://www.aircominternational.com/Schemas/
                   CommonTypes/2013/03"/>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

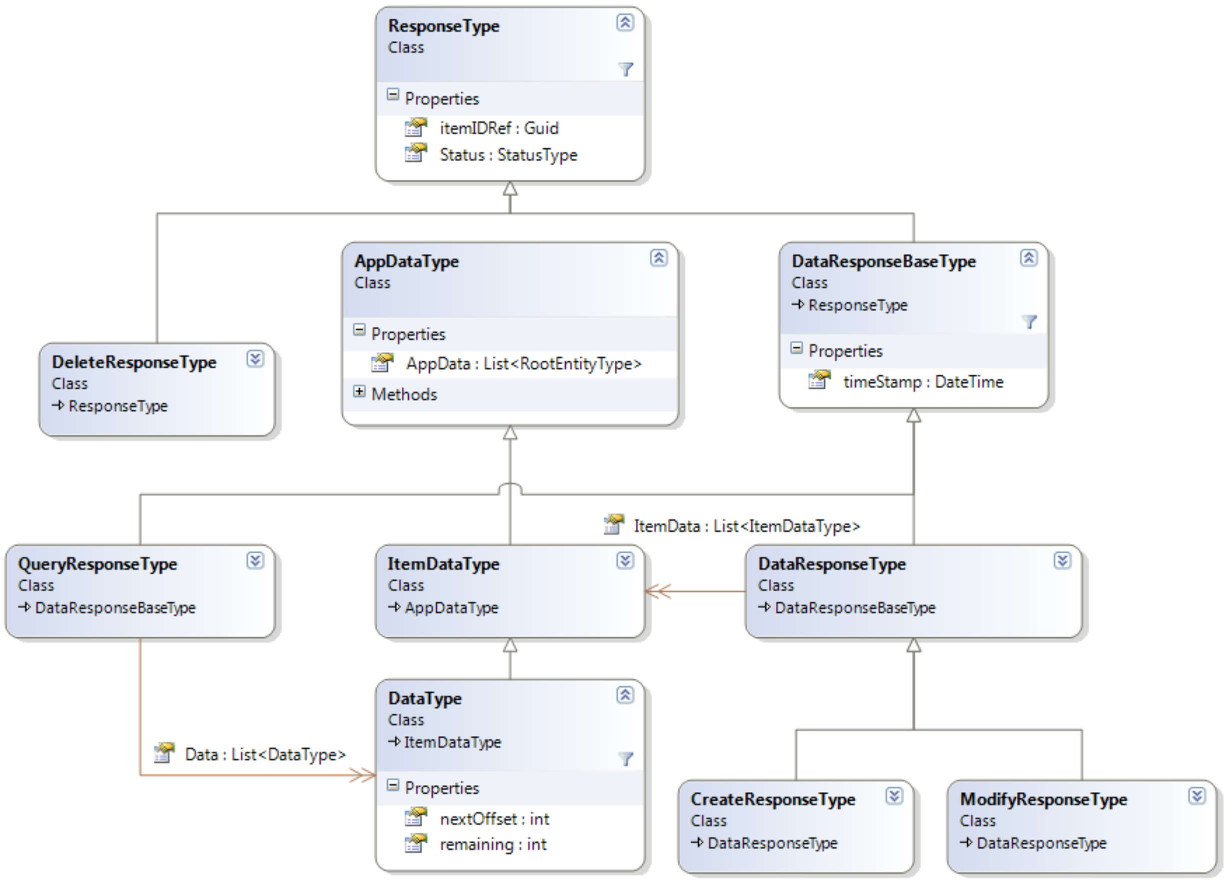
The schema extract for ListNbrItemType is:

```
<xs:complexType name="ListNbrItemType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:ListItemType">
                  <xs:attribute name="NbrTechDiscriminator"
type="q5:NbrTechDiscriminatorType"
xmlns:q5="http://www.aircominternational.com/Schemas/CommonTypes/2013/03"
/>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

Responses

Each of the operations (Create, Query, Update, and Delete) must return a response meeting the requirements of the request and embedded request items. Reciprocally, the structure of the response contracts contains sub-elements in order to pair with the request items.

Additionally, each response will contain Status elements which indicate the overall and partial success of the request; this element also includes any error detail raised by the service. This picture shows the structure of the responses:



Response structure in the EDS

All four responses share a common base type of ResponseType, but the Create and Modify responses are derived from a DataResponseType in order to support the ResultQuery if supplied by their request. The QueryResponseType does not share this base as its collection, although through the Data property will also indicate where the data has come from inside the rowset.

Note: The collection's Data and ItemData are all essentially lists of AppDataType on page 36. The DataType class provides an extension to this type by indicating from the query it was asked to execute how many rows are remaining and where the next offset begins.

This table summarizes the properties within the response hierarchy:

Property Name	Type	Description
itemIDRef	GUID	Matching ID to the request which the client supplied used to correlate a request to a response.
Status	StatusType	An attached object which indicates the success of the overall and sub components of the request items.
Data	List<DataType>	A collection of DataType objects, produced by a query response.
ItemData	List<ItemDataType>	Similar to Data, but does not include any pagination information. It is intended that a ResultQuery will focus on the object which was just modified, rather than ask for a fully built rowset.
nextOffset	int	An indication that if you want to page the rowset, you should start from this position.
remaining	int	An indication as to how many rows are remaining in the rowset.
timeStamp	DateTime	A timestamp indicating when the response was sent.

Response Type Schemas

ResponseType

The schema extract for ResponseType is:

```
<xs:complexType name="ResponseType">
      <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="1" name="Status"
type="tns:StatusType"/>
      </xs:sequence>
      <xs:attribute  name="itemIDRef" type="q3:guid" use="required"
       xmlns:q3="http://microsoft.com/wsdl/types/"/>
</xs:complexType>
```

DataResponseBaseType

The schema extract for DataResponseBaseType is:

```
<xs:complexType name="DataResponseBaseType">
      <xs:complexContent mixed="false">
            <xs:extension base="q3:ResponseType"
xmlns:q3="http://www.aircominternational.com/contract/Util/2009/10">
                  <xs:attribute name="timeStamp" type="xs:dateTime"
use="required"/>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

DataResponseType

The schema extract for DataResponseType is:

```
<xs:complexType name="DataResponseType">
      <xs:complexContent mixed="false">
            <xs:extension base="q18:DataResponseBaseType"

xmlns:q18="http://www.aircominternational.com/contract/EDS/DST/2009/10">
                  <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="unbounded"
                         name="ItemData" type="tns:ItemDataType"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

DeleteResponseType

The schema extract for DeleteResponseType is:

```
<xs:complexType name="DeleteResponseType">
      <xs:complexContent mixed="false">
            <xs:extension base="q19:ResponseType"

xmlns:q19="http://www.aircominternational.com/contract/Util/2009/10"/>
      </xs:complexContent>
</xs:complexType>
```

CreateResponseType

The schema extract for CreateResponseType is:

```
<xs:complexType name="CreateResponseType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:DataResponseType"/>
      </xs:complexContent>
</xs:complexType>
```

ModifyResponseType

The schema extract for ModifyResponseType is:

```
<xs:complexType name="ModifyResponseType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:DataResponseType"/>
      </xs:complexContent>
</xs:complexType>
```

QueryResponseType

The schema extract for QueryResponseType is:

```
<xs:complexType name="QueryResponseType">
      <xs:complexContent mixed="false">
            <xs:extension base="q21:DataResponseBaseType"
             xmlns:q21="http://www.aircominternational.com/contract/EDS/
             DST/2009/10">
                  <xs:sequence>
```

```
                        <xs:element minOccurs="0" maxOccurs="unbounded"
                         name="Data" type="tns:DataType"/>
                  </xs:sequence>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

ItemDataType

The schema extract for ItemDataType is:

```
<xs:complexType name="ItemDataType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:AppDataType">
                  <xs:attribute ref="q10:itemIDRef"

xmlns:q10="http://www.aircominternational.com/contract/Util/2009/10"/>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

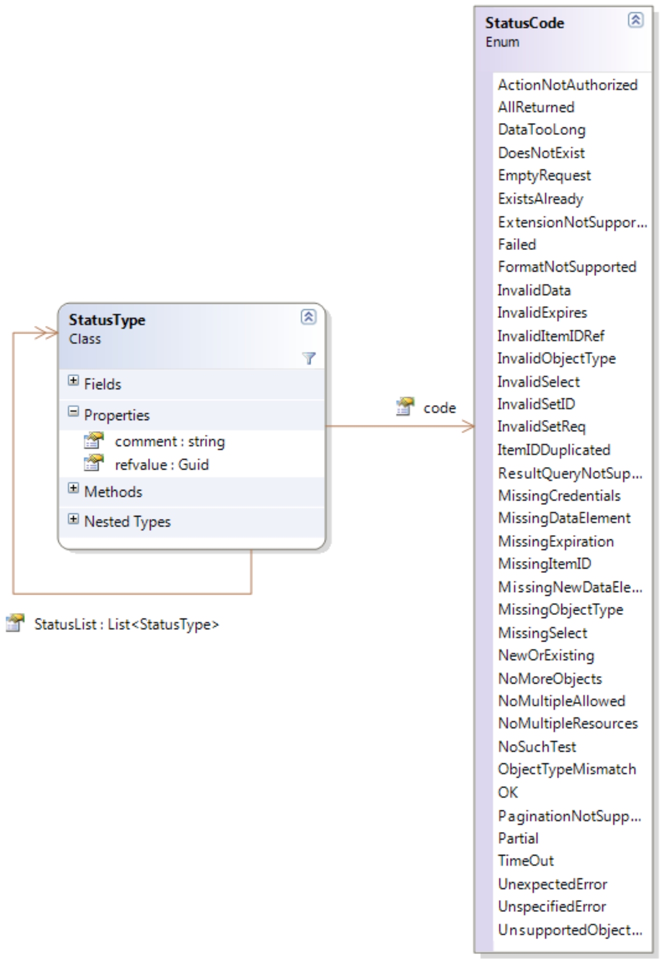
DataType

The schema extract for DataType is:

```
<xs:complexType name="DataType">
      <xs:complexContent mixed="false">
            <xs:extension base="tns:ItemDataType">
                  <xs:attribute
                   name="remaining" type="xs:int" use="required"/>
                  <xs:attribute
                   name="nextOffset" type="xs:int" use="required"/>
            </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

ResponseStatus

The status type is a composite type which allows responses to contain an outer overall status of the request and individual statuses which map to the request items. This picture shows the structure of the StatusType:



Structure of Response Status Types in the EDS

This table describes the properties of the StatusType:

Property Name	Type	Description
refvalue	GUID	Matching id to the request which was supplied by the client. If this is the outer StatusType, then the refvalue correlates to the Request itemID. If it is an embedded status item it correlates to the RequestItem.
Code	StatusCode	An enumeration representing all possible values which the EDS might return. The outer status item will always be one of these values: • OK - the request and all request items executed correctly • Partial - some of the request items failed to execute. Check the embedded status responses • Failed - the request and all of its embedded items failed to execute. Check the comments field for details
comment	String	A textual error message indicating more details regarding a failure.
StatusList	List<StatusType>	A collection of sub-status items used to correlate against individual request items.

Status Type XML Schema

The schema for StatusType is:

```
<xs:complexType name="StatusType">
      <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded"
             name="Status" type="tns:StatusType" />
            <xs:element minOccurs="0" maxOccurs="1" name="comment"
type="xs:string" />
      </xs:sequence>
      <xs:attribute name="code" use="required">
            <xs:simpleType>
                  <xs:restriction base="xs:string">
                        <xs:enumeration value="ActionNotAuthorized" />
                        <xs:enumeration value="AllReturned" />
                        <xs:enumeration value="DataTooLong" />
                        <xs:enumeration value="DoesNotExist" />
                        <xs:enumeration value="EmptyRequest" />
                        <xs:enumeration value="ExistsAlready" />
                        <xs:enumeration value="ExtensionNotSupported" />
                        <xs:enumeration value="Failed" />
                        <xs:enumeration value="FormatNotSupported" />
                        <xs:enumeration value="InvalidData" />
                        <xs:enumeration value="InvalidExpires" />
                        <xs:enumeration value="InvalidItemIDRef" />
                        <xs:enumeration value="InvalidObjectType" />
                        <xs:enumeration value="InvalidSelect" />
                        <xs:enumeration value="InvalidSetID" />
                        <xs:enumeration value="InvalidSetReq" />
                        <xs:enumeration value="ItemIDDuplicated" />
                        <xs:enumeration value="ResultQueryNotSupported"
/>
                        <xs:enumeration value="MissingCredentials" />
                        <xs:enumeration value="MissingDataElement" />
                        <xs:enumeration value="MissingExpiration" />
                        <xs:enumeration value="MissingItemID" />
                        <xs:enumeration value="MissingNewDataElement" />
                        <xs:enumeration value="MissingObjectType" />
                        <xs:enumeration value="MissingSelect" />
                        <xs:enumeration value="NewOrExisting" />
                        <xs:enumeration value="NoMoreObjects" />
                        <xs:enumeration value="NoMultipleAllowed" />
                        <xs:enumeration value="NoMultipleResources" />
                        <xs:enumeration value="NoSuchTest" />
                        <xs:enumeration value="ObjectTypeMismatch" />
                        <xs:enumeration value="OK" />
                        <xs:enumeration value="PaginationNotSupported" />
                        <xs:enumeration value="Partial" />
                        <xs:enumeration value="TimeOut" />
                        <xs:enumeration value="UnexpectedError" />
                        <xs:enumeration value="UnspecifiedError" />
                        <xs:enumeration value="UnsupportedObjectType" />
                        <xs:enumeration value="Warning" />
                  </xs:restriction>
            </xs:simpleType>
      </xs:attribute>
      <xs:attribute xmlns:q5="http://microsoft.com/wsdl/types/"
       name="ref" type="q5:guid" use="required" />
      <xs:attribute xmlns:q6="http://microsoft.com/wsdl/types/"
       name="masterIDRef" type="q6:guid" />
</xs:complexType>
```

EDS Basic Web Service Interface

This section deals with the service contract and objects that make up the primary interface for manipulating data with the EDS.

The base endpoint address for this interface (WS-HTTP) is:

```
http://{server}:{port}/Aircom/EDSBasic/WS
```

A Microsoft-compatible “netTCP” endpoint is:

```
netTCP://{server}:8634/Aircom/EDSBasic/TCP
```

Its WSDL can be retrieved from this URI:

```
http://{server}:8630/Aircom/EDSBasic/WS?wsdl
```

EDS Basic Service Contract Architecture

The EDS Basic interface has a simplified contract architecture to help reduce the amount of code that is produced by its WSDL definition, but still follows the same architecture as the original.

Its service contract architecture is almost identical to that of the Strongly Typed interface however it does not supply data types describing the ENTERPRISE data model within its WSDL definition. This is useful when you want to create a more minimal client integration, or whereby your client cannot correctly handle the data type stubs necessary to use the typed interface. Furthermore effort has been made to streamline the use of attribute, or external namespace declarations within the XML signature of this interface to further simplify client interaction.

In place of requiring strongly typed data when serving requests, the EDS Basic interface uses an opaque XML stream where ENTERPRISE element data is passed directly rather than using the code classes produced by the WSDL as in the original service.

XML validation is done using the appropriate schema in the samples folder of the installed product.

The service contract uses a principle known “as operations as data”, and therefore the EDS only offers four basic methods for data manipulation that all accept one parameter:

- CREATE
- UPDATE
- QUERY
- DELETE

Each request contains facets which describe the data to be queried or manipulated, as well as additional metadata which controls how the service will behave in response to the request. For example, the query request will contain metadata describing the query, as well as paging parameters to indicate which part of the result set to return.

Contract Implementation

A simplified view of the service interface is provided in this picture to illustrate the operation signature of the methods described in EDS Basic Service Contract Architecture on page 49.

All operations use XmlSerializationFormat.

```
[ServiceContract(Namespace = BasicNamespaceTypes.tns, Name = "EDSBasic")]
public interface IEDSBasicService
{
```

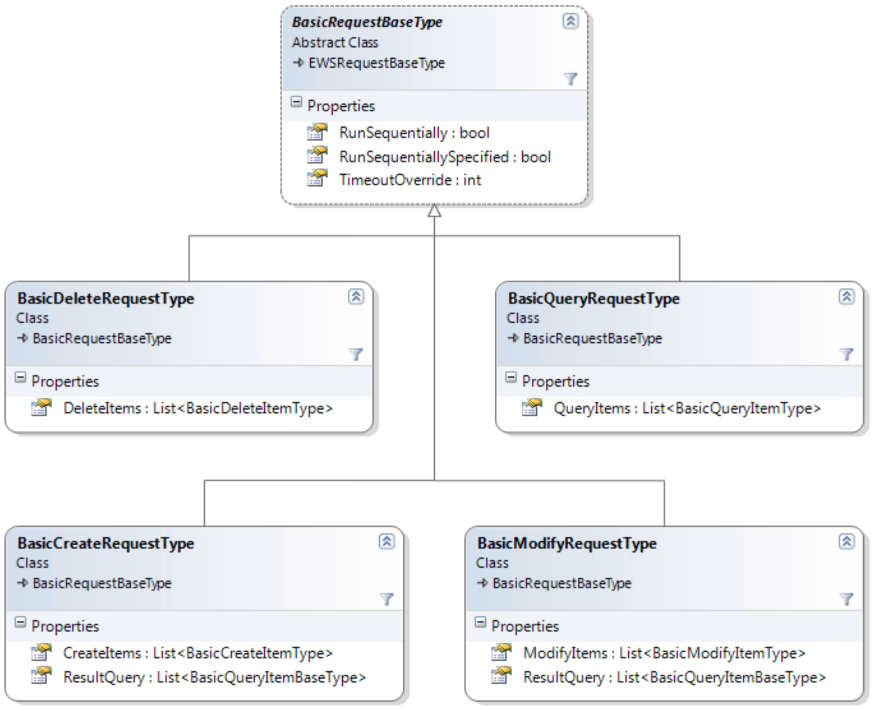
```
                [OperationContract]
                BasicCreateResponseType Create (BasicCreateRequestType data);

                [OperationContract]
                BasicQueryResponseType Query (BasicQueryRequestType data);

                [OperationContract]
                BasicModifyResponseType Modify (BasicModifyRequestType data);

                [OperationContract]

                BasicDeleteResponseType Delete (BasicDeleteRequestType data);

                [OperationContract]
                SupportedTypes QueryableTypes ();

                [OperationContract]
                SupportedTypes WriteableTypes ();

}
```

The two ancillary methods on the service contract perform the following:

- QueryableTypes - returns an enumeration indicating all available data types within the EDS
- WriteableTypes - a subset of the previous enumeration

Each request operation shares a common request base type, and elements which are specific to that operation.

This picture shows the hierarchy for these request types:



BasicRequest Type Hierarchy

Each request type has a unique ItemID used to correlate the message through the system, a collection of sub-items which allow for batching of multiple data types within a single request, and, depending on the operation, a ResultQuery which allows for the inclusion of a query or queries after the data manipulation has completed.

In all other respects, the information is the same as for EDS Strongly Typed Service Contract Architecture.

3 Batch Processing Using the EDS

When processing large datasets within the EDS Web Service Interface, the following features can help to make this more efficient:

- RunSequentially – Instructs EDS to execute all “RequestItems”, within a single request serially rather than concurrently.
- FileItem – Request from EDS an XML file containing your data.
- InputFile – Supply EDS with an XML file containing your data.
- FileLoadType – A “dummy” object type for passing large amounts of XML data more efficiently.

Note: You will need to set an appropriate timeout to correctly use these features. Given that you are potentially providing EDS with more data to process, you will need to set a larger interval. You may also need to structure the size of the data volume more appropriately to ensure continuous throughput without timeout.

RunSequentially

“RunSequentially” is an optional parameter that allows the user to specify that all operations contained within a single “request” take place in the same thread. By default (in a single process installation) EDS will run up to 8 threads to process data throughput. Each ModifyItems, CreateItems, DeleteItems and QueryItems entry will usually run within a different thread. For example, if a large request contains 50 ModifyItems then they will be spread across all threads to attempt to speed the data throughput. This is not necessarily a good way to utilise EDS depending on your server configuration, as it may cause unnecessary database thrashing while each thread competes for database access.

Setting RunSequentially to true will force EDS to run each of the ModifyItems requests consecutively in the same thread.

This means that a user can pass different types of data in a single request and ensure the order in which they are processed, such as adding LocationObjectTypes before CellSiteTypes, and CellSiteTypes before GSMCellTypes.

FileItem

This can only be used for QueryItems and is used to write large datasets to a passed in directory location to receive the XML data rather than stream its entire payload XML via its interface.

```
<ns:QueryItems ns:objectType="LocationObjectType" exclusion="Security"
 count="100" offset="0">
      <ns1:itemID>C7B970A6-E0A4-4782-973D-3835AE656F77</ns1:itemID>
      <ns:Select>
            <ns4:Simple>
                  <ns4:OverrideCacheDefault UseQueryCache="false"/>
                  <ns4:Query>bvid="Bob"</ns4:Query>
            </ns4:Simple>
      </ns:Select>
      <ns:FileItem>
            <URI>\\MyServer\MyShare</URI>
            <Impersonate>
                  <Domain>bob</Domain>
                  <User>brian.oliver</User>
                  <Password>bob</Password>
            </Impersonate>
      </ns:FileItem>
</ns:QueryItems>
```

Note: The URI element must be a fully qualified path to a directory; a filename should not be passed.

InputFile

This can only be used for DeleteItems and is used to read large datasets from a passed in file location to read the XML data from rather than stream its entire payload XML via its interface.

```
<ns:DeleteItems>
      <ns1:itemID>D7B970A6-E0A4-4782-973D-3835AE656F77</ns1:itemID>
      <ns:objectType>LocationObjectType</ns:objectType>
      <ns:Select>
            <ns4:InputFile>
                  <ns4:URI>\\MyServer\MyShare</ns4:URI>
                  <ns4:Impersonate>
                        <ns4:Domain>adomain </ns4:Domain>
                        <ns4:User>user</ns4:User>
                        <ns4:Password>password</ns4:Password>
                  </ns4:Impersonate>
            </ns4:InputFile>
      </ns:Select>
</ns:DeleteItems>
```

The **URI** element is either a fully qualified path to an EDS XML file, or a fully qualified path to a directory containing one or more EDS XML files.

If a single file has been supplied then EDS will process just that file, but if a directory has been supplied then all files of .XML type will be processed from that directory.

For both FileItem and InputFile the **URI** is followed by an **Impersonate** element containing **Domain**, **User** and **Password**.

This is used when the application does not have access to a file or directory, like a file on a server, in which case a valid domain name, user name and password must be supplied that has access to that file/directory.

This file processing method can be used on **QueryItems** via FileItem element, **DeleteItems** via the InputFile element and **ModifyItems** and **CreateItems** within the **FileLoadType** (as described in the next section).

Security Note: By default the EDS SOAP and REST interfaces are using an **unencrypted** HTTP socket for transmission of data. If you plan on sending access credentials to EDS in order to manipulate files you should consider configuring EDS to use HTTPS transport level encryption. Alternatively you can grant the EDS “Presentation Service” component permission to access a remote file location via your Windows application server using the “Service (services.msc)” control panel. If you would like more information on using either method of securely file access then please contact Support.

FileLoadType

A “dummy” object type called **FileLoadType** has been created as a means of passing a URI (file path) to the persistence service. This makes passing large amounts of XML data more efficient as the payload only contains a few hundred bytes, and can resemble the following:

<ns:ModifyItems ns:objectType="LocationObjectType" ns:SparseListUpdate="false" ns:CreateIfNotFound="true" ns:IgnoreQueryAndDoBulkUpdate="true">

```
      <ns1:itemID>C7B970A6-E0A4-4782-973D-3835AE656F79</ns1:itemID>
      <ns:NewData>
            <ns:AppData xsi:type="co81:FileLoadType" ns5:bvid="UK">
                  <co81:FileItem>

      <URI>\\MyServer\MyShare\LocationObjecttypes.xml</URI>
                        <Impersonate>
                              <Domain>adomain</Domain>
                              <User>user</User>
                              <Password>pass</Password>
                        </Impersonate>
                  </co81:FileItem>

      <co81:AllowedOperations>ReadWrite</co81:AllowedOperations>
            </ns:AppData>
      </ns:NewData>
</ns:ModifyItems>
```

This type is ignored by the other operations and cannot be queried or deleted, it is only available to the **ModifyItem** and the **CreateItem** request types.

The FileLoadType uses another new query item, **FileItem** as described above.

In this example the LocationObjectType has been specified as the **objectType**, which instructs EDS to only process objects of this type. So if a directory is supplied, EDS will read all .XML files but will only process objects of supplied type and ignore any others. So if you supply an **objectType** of LocationObjectv81Type but then supply a file containing LocationObjectTypes then the LocationObjectTypes will be ignored.

Attributes:

IgnoreQueryAndDoBulkUpdate="?":

IgnoreQueryAndDoBulkUpdate=”false” is not a valid option as this is a bulk update and setting this to false will result in an error.

SparseListUpdate="?" and **CreateIfNotFound**="?":

Will operate as normal.

bvid=”?”:

If **bvid** is supplied then all of the **eids** (object’s DB key) will be removed from the input and the project will be replaced with bvid specified project (the same functionality as the EDS loader).

4 About the EDS REST Interface

In this release EDS supports two RESTful interfaces:

- A legacy REST interface that provides read-only support to the same datatypes as the SOAP interface and is a lightweight means of testing or debugging issues with the service; however, it can also be used for application development. The only other significant difference is that whilst the EDS Web Service interfaces use a buffered MTOM over the wire format for efficiency, the REST interface will always return plain text XML.
- A fully RW REST interface based upon the OpenAPI standard that supports authentication via declared authorisation keys held within the EDS configuration. It is intended that this interface will replace the legacy variant in a later release. Furthermore, this interface accepts EDS strongly typed XML format or JSON formatted data types.

Legacy REST Interface

All commands are expressed over the URL as query string parameters, with the exception of the object type which forms part of the URI.

The signature of the REST requests supported by the EDS looks like this example:

```
http://{server}:{port}/Aircom/EDS/REST/{type}/?qry={query}&row={row}
&max={max}&export={export}&timeout={TimeoutOverride}&cache={OverrideCache
}
&exclude={sub-elements}&include={sub-elements}
```

This table describes the parameters:

Parameter	Description.
Host	The base address of the server hosting the service.
Type	The object type you want to query.
Qry	A query string which supports the simplified text-based query notation.
Row	Starting position for the EDS to fetch data from within the rowset returned by the query.
Max	Maximum number of rows returned in a single request. The EDS default is set to 10.
Export	You can optionally export the result to a disk location rather than passing it directly back to the client.
Timeout	An override to the default 45 second timeout within the EDS
Cache	Instructs the EDS not to use object caching. This is useful when you want the most up-to-date copy, otherwise this could cause a negative performance hit.
Exclude	Instructs EDS to ignore certain sub-elements within the specified type. This is useful for creating sparse data, which can significantly improve performance.
Include	Instructs EDS to include only certain sub-elements within the specified type. This is useful for creating sparse data, which can significantly improve performance.

Note: If you do not supply any query parameters, the EDS will return the first 10 rows of data for the indicated type within the URI.

This is an example of a typical result set from the REST interface:

```
<QueryResponseType xmlns:ct="http://www.aircominternational.com/Schemas/
CommonTypes/2009/05" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns:gsm="http://www.aircominternational.com/Schemas/GSM/2009/09"
xmlns:eqp="http://www.aircominternational.com/Schemas/Equipment/2009/09"
xmlns:co="http://www.aircominternational.com/Schemas/Common/2009/07"
xmlns:umts="http://www.aircominternational.com/Schemas/UMTS/2009/05"
xmlns:util="http://www.aircominternational.com/contract/Util/2009/10"
xmlns:tra="http://www.aircominternational.com/Schemas/Connect/2009/09"
util:itemIDRef="87927e23-59dd-4f53-a9fb-cfc2459211ed"
timeStamp="2010-03-17T20:42:37.8654434+00:00"
xmlns="http://www.aircominternational.com/contract/EDS/2009/05">
      <util:Status code="OK" ref="87927e23-59dd-4f53-a9fb-cfc2459211ed">
            <util:Status code="OK" ref="3a0383f4-29d8-446e-99b1-
619061d96767" />
      </util:Status>
      <Data util:itemIDRef="3a0383f4-29d8-446e-99b1-619061d96767"
       remaining="169262" nextOffset="10">
            <AppData xsi:type="co:LocationObjectType" ct:iid="NT05223C"
             ct:bvid="Northeast" ct:eid="574810425">
                  <Security xmlns="">
                        <CreateDate>2009-04-27T17:12:07</CreateDate>
                        <ModifyDate>2010-01-20T17:33:25</ModifyDate>
                  </Security>
                  <CustomFields xmlns="">
                        <Field Group="Candidate Status" Value="Candidate"
/>
                        <Field Group="FCC_MW Status Flags" Value="N/A" />
                        <Field Group="Special Circumstance" />
                        <Field Group="I_Lat_Long Method" Value="GPS" />
                        <Field Group="I_Non Standard Site" Value="N/A" />
                        <Field Group="I_Microwave Only Site" Value="No"
/>
                        <Field Group="RFDS_TotAntSector1" Value="N/A" />
                        <Field Group="RFDS_TotAntSector2" Value="N/A" />
                        <Field Group="RFDS_TotAntSector3" Value="N/A" />
                        <Field Group="RFDS_TotAntSector4" Value="N/A" />
                        <Field Group="RFDS_TotCoaxSector1" Value="N/A" />
                        <Field Group="RFDS_TotCoaxSector2" Value="N/A" />
                        <Field Group="RFDS_TotCoaxSector3" Value="N/A" />
                        <Field Group="RFDS_TotCoaxSector4" Value="N/A" />
                        <Field Group="FSC_Geo_Mkt" Value="New Jersey, NJ"
/>
                  </CustomFields>
                  <co:Address1>1092 Cassini Court</co:Address1>
                  <co:Town>Leatherhead</co:Town>
                  <co:State>NJ</co:State>
                  <co:PostCode>07111</co:PostCode>
```

Note: The result set does not remove the containing query response or status of the request.

OpenAPI R/W REST Interface

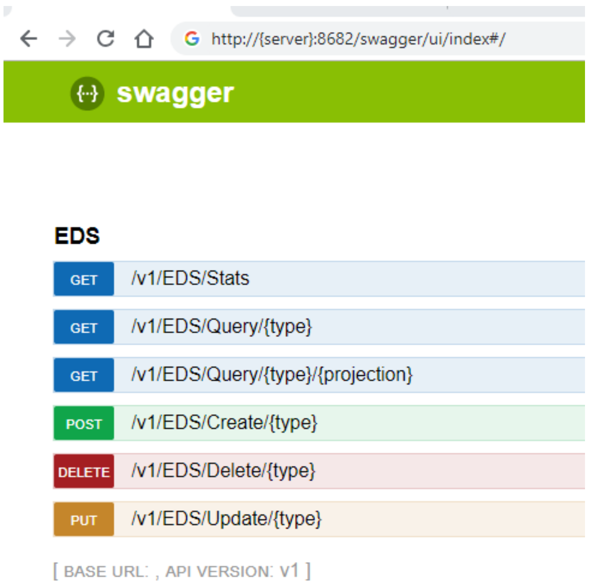
This interface provides the same R/W capability as the native EDS SOAP API but adds the convenience of adding the features of EDS to more Web based clients. It is anticipated that application integrators would still use the SOAP interface for more bulk volume processing and this endpoint for more targeted use cases (that is, specific data type updates, simple HTML visualisations of ASSET data and so on).

The new endpoint is available on the following port, this can be changed via the EDS configuration file:

```
http://{server}:8682
```

Swagger UI

The OpenAPI standard includes a built-in mechanism to self-document its RESTful interfaces, by default if you enter the above URL (where {server} is the name of your EDS server) you will be presented with the following web page:



EDS

GET	/v1/EDS/Stats
GET	/v1/EDS/Query/{type}
GET	/v1/EDS/Query/{type}/{projection}
POST	/v1/EDS/Create/{type}
DELETE	/v1/EDS/Delete/{type}
PUT	/v1/EDS/Update/{type}

[BASE URL: , API VERSION: v1]

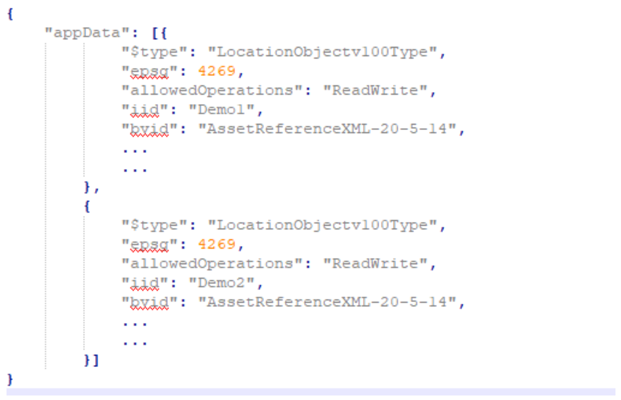
From this user interface you can then navigate to the various supported RESTful methods that EDS offers and experiment or test with EDS data as you require. The interface will indicate the appropriate optional or mandatory parameters that the methods require and accept pasted input from the clipboard when requiring EDS data types.

Note: These methods are identical to those offered by the SOAP interface and will function in a similar way.

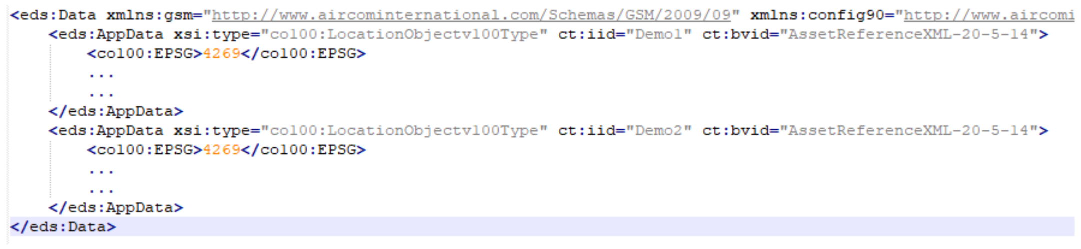
Differences Between the RESTful Interface and the SOAP Interface

Although this REST interface offers the same methods and parameter control as the SOAP interface, there are a few optimisations and simplifications of the EDS Request or Response structures in order to streamline integration and application development with the interface:

- No Request/RequestItem structures required – Create/Update/Delete operations just require a simple object containing an appData array of (a singular) EDS datatype. The following provides examples in JSON and XML format:

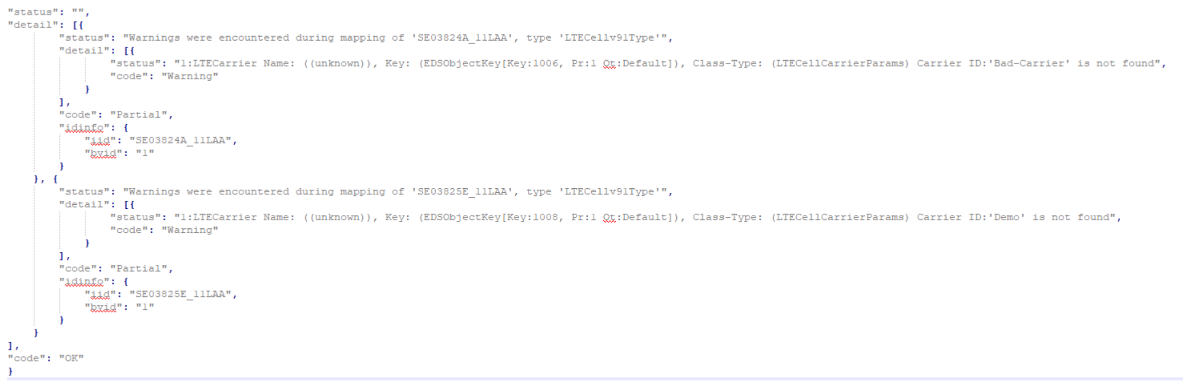


```
{
    "appData": [{
            "$type": "LocationObjectv100Type",
            "epsg": 4269,
            "allowedOperations": "ReadWrite",
            "iid": "Demo1",
            "bvid": "AssetReferenceXML-20-5-14",
            ...
            ...
        },
        {
            "$type": "LocationObjectv100Type",
            "epsg": 4269,
            "allowedOperations": "ReadWrite",
            "iid": "Demo2",
            "bvid": "AssetReferenceXML-20-5-14",
            ...
            ...
        }]
}
```



```
<eds:Data xmlns:gsm="http://www.aircominternational.com/Schemas/GSM/2009/09" xmlns:config90="http://www.aircomi
    <eds:AppData xsi:type="co100:LocationObjectv100Type" ct:iid="Demo1" ct:bvid="AssetReferenceXML-20-5-14">
        <co100:EPSG>4269</co100:EPSG>
        ...
        ...
    </eds:AppData>
    <eds:AppData xsi:type="co100:LocationObjectv100Type" ct:iid="Demo2" ct:bvid="AssetReferenceXML-20-5-14">
        <co100:EPSG>4269</co100:EPSG>
        ...
        ...
    </eds:AppData>
</eds:Data>
```

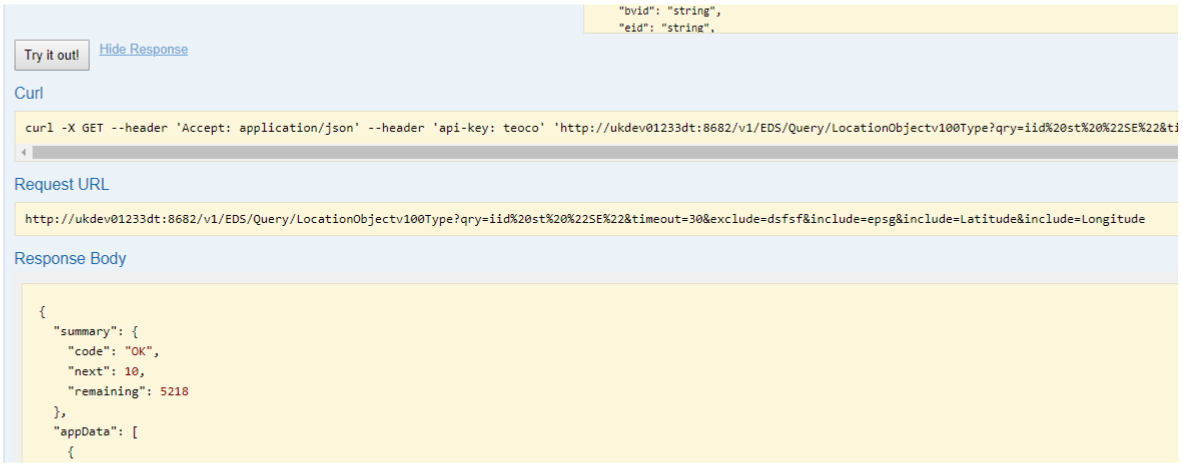
- Flattened EDS status summary info – The status information has been partially flattened to make it easier to traverse nested warnings:



```
"status": "",
"detail": [{
        "status": "Warnings were encountered during mapping of 'SE03824A_11LAA', type 'LTECellv91Type'",
        "detail": [{
                "status": "1:LTECarrier Name: ((unknown)), Key: (EDSObjectKey[Key:1006, Pr:1 Qt:Default]), Class-Type: (LTECellCarrierParams) Carrier ID:'Bad-Carrier' is not found",
                "code": "Warning"
            }
        ],
        "code": "Partial",
        "idinfo": {
            "iid": "SE03824A_11LAA",
            "bvid": "1"
        }
    }, {
        "status": "Warnings were encountered during mapping of 'SE03825E_11LAA', type 'LTECellv91Type'",
        "detail": [{
                "status": "1:LTECarrier Name: ((unknown)), Key: (EDSObjectKey[Key:1008, Pr:1 Qt:Default]), Class-Type: (LTECellCarrierParams) Carrier ID:'Demo' is not found",
                "code": "Warning"
            }
        ],
        "code": "Partial",
        "idinfo": {
            "iid": "SE03825E_11LAA",
            "bvid": "1"
        }
    }
],
"code": "OK"
}
```

Using the UI for Testing and Prototyping

The CRUD methods in the UI all provide a simple explanation of the syntax available on the RESTful interface and will construct an appropriate HTTP command for using the interface when the user clicks the “Try it out!” button at the bottom of the screen. For example the following is from an EDS query for a v100 LocationObject with some simple include and EDS text query syntax:



```
"bvid": "string",
"eid": "string",
```

```
curl -X GET --header 'Accept: application/json' --header 'api-key: teoco' 'http://ukdev01233dt:8682/v1/EDS/Query/LocationObjectv100Type?qry=iid%20st%20%22SE%22&ti
```

```
http://ukdev01233dt:8682/v1/EDS/Query/LocationObjectv100Type?qry=iid%20st%20%22SE%22&timeout=30&exclude=dsfsf&include=epsg&include=Latitude&include=Longitude
```

```
{
  "summary": {
    "code": "OK",
    "next": 10,
    "remaining": 5218
  },
  "appData": [
    {
```

For Query methods you can also take the “Request URL” string provided and see the data independently using any web browser. For EDS Update/Delete/Create methods that require a non-GET verb (HTTP PUT/POST/DELETE respectively). Swagger also offers a command line syntax via an independent utility named “curl” that can be downloaded freely for Windows or is included as part of the Linux OS.

Note: In the above example, EDS will still provide “next” and “remaining” row count indicators to help navigate paginated data sets.

Working with XML or JSON Data Sets

The RESTful interface accepts or responds with either EDS XML or new lightweight JSON formatted objects to describe EDS data types but it is necessary to tell the interface which format you want it to use via the following:

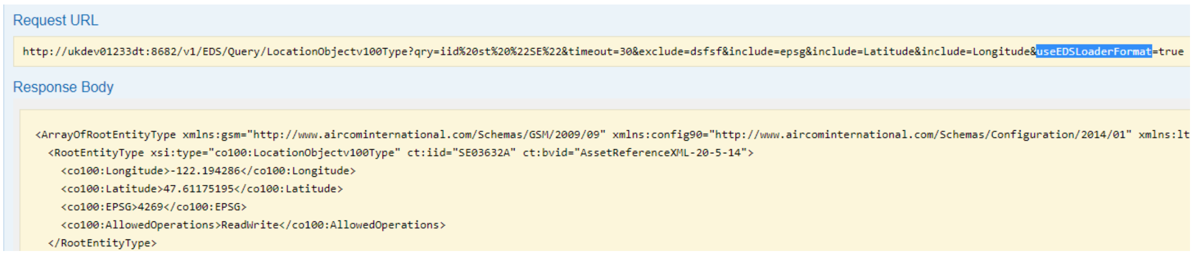
- Query – Either include an optional “format=JSON|XML” query parameter to the request or set an HTTP request header to: “Accept: application/xml”
- Create/Update/Delete – Both the HTTP “Accept” and “Content-Type” headers need to be set to “application/xml” or “application/json” for XML or JSON content.

Generating EDS Loader Format XML

EDS loader is a command line utility available with EDS that is useful for bulk processing or testing the EDS interface.

The RESTful Query command provides an option named “useEDSLoaderFormat” that will generate an EDS Loader compliant XML response that can then be saved and imported using EDS Loader. If you look closely, the generated XML contains the appropriate EDS Loader header element and omits the “Summary” information that would normally contain error info and pagination data for the request.

Note: This XML is not compatible with the format accepted by the RESTful Create/Update/Delete methods.


Request URL

```
http://ukdev01233dt:8682/v1/EDS/Query/LocationObjectv100Type?qry=iid%20st%20%22SE%22&timeout=30&exclude=dsfsf&include=epsg&include=Latitude&include=Longitude&useEDSLoaderFormat=true
```

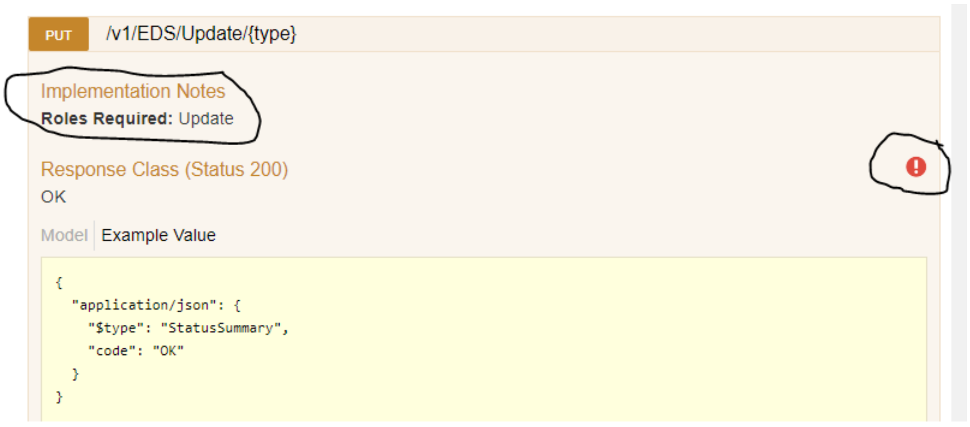
Response Body

```
<ArrayOfRootEntityType xmlns:gsm="http://www.aircominternational.com/Schemas/GSM/2009/09" xmlns:config90="http://www.aircominternational.com/Schemas/Configuration/2014/01" xmlns:lt
  <RootEntityType xsi:type="co100:LocationObjectv100Type" ct:iid="SE03632A" ct:bvid="AssetReferenceXML-20-5-14">
    <co100:Longitude>-122.194286</co100:Longitude>
    <co100:Latitude>47.61175195</co100:Latitude>
    <co100:EPSG>4269</co100:EPSG>
    <co100:AllowedOperations>ReadWrite</co100:AllowedOperations>
  </RootEntityType>
```

Securing the RESTful Interface with API-keys

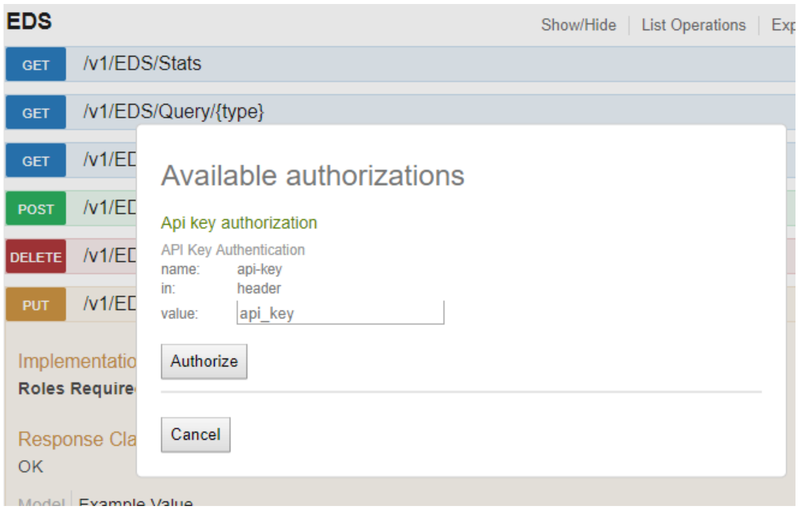
By default, the RESTful interface will not permit any user who has not supplied an appropriate API key to the interface, only the Query and Stats methods can be used anonymously.

From the Swagger UI, if you look more closely at the secured methods you may note that each has a required role and a little red exclamation mark button that you can interact with:



```
{
  "application/json": {
    "$type": "StatusSummary",
    "code": "OK"
  }
}
```

If you select the red button, the UI will prompt you for an API-key. This value will then be cached by the Swagger UI until you refresh or close your web browser.



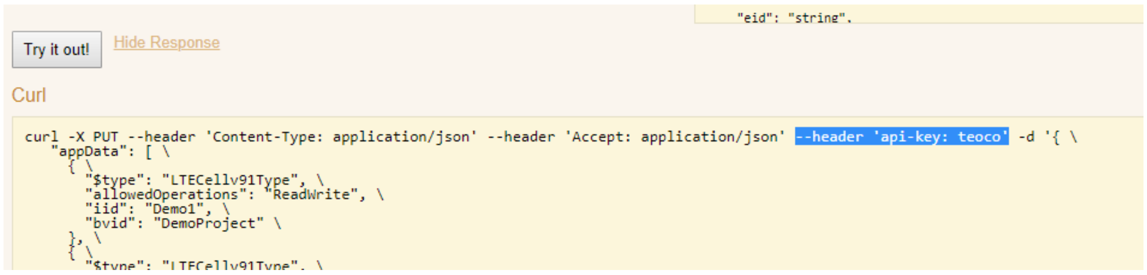
name:	api-key
in:	header
value:	api_key

From this prompt you can enter a provided key and click the “Authorize” button to set it. By default, EDS includes a singular API key named “teoco” that can be used for testing but should be removed from the EDS config before your API is put into production.

Once the key has been established, the UI will change the colour of the exclamation to blue and allow you to use the “Try it out!!” button with the Create/Update/Delete methods.

Setting the API-key from Code

The API key is implemented as another HTTP header value named “api-key”, you can see an example of it in the curl syntax that the UI offers:

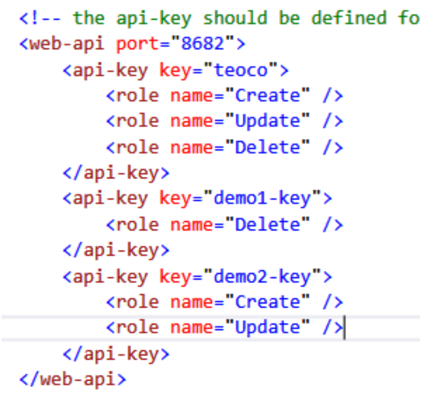


```
                                                            "eid": "string",
```

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'api-key: teoco' -d '{ \
   "appData": [ \
     { \
       "$type": "LTECellv91Type", \
       "allowedOperations": "ReadWrite", \
       "iid": "Demo1", \
       "bvid": "DemoProject" \
     }, \
     { \
       "$type": "LTECellv91Type", \
```

Configuring API-keys and Assigning Roles

API keys and their role mapping is configured via the EDS application config files available (either TEOCO.EDS.Presentation.Runtime.config or …Composite.Runtime.config):



```
<!-- the api-key should be defined fo
<web-api port="8682">
    <api-key key="teoco">
        <role name="Create" />
        <role name="Update" />
        <role name="Delete" />
    </api-key>
    <api-key key="demo1-key">
        <role name="Delete" />
    </api-key>
    <api-key key="demo2-key">
        <role name="Create" />
        <role name="Update" />
    </api-key>
</web-api>
```

The API key may be any valid string (without spaces or reserved characters) and then its mapping to appropriate EDS commands is nested beneath it. In the above example you can see the default “teoco” key and then two more named “demo1-key” and “demo2-key”.

Note: If you assign or need to change an API key or its mapping you must restart EDS.

5 EDS Simplified Query Notation

Queries to the EDS can be created either by using a structured XML document describing the query, or by using a simplified freeform text format. This section describes the keywords which can be used to interact with this format.

All queries are performed against a supported data type and you cannot query over multiple types within the same request to the Web Service. Freeform queries follow the following basic syntax and rules:

`{type}` <attribute> <operation> <value>

Note: Within the API "type" is specified using a parameterised value which is specific to the interface. It has been included here to aid explanation of the freeform text query language.

Here is an example of a freeform query:

```
{gsmcell} iid = “TestID”
```

Where "type" would be a GSM cell and the query is looking for a matching element with an ID of "TestID".

Queries can also be combined using a predicate to offer combination of attribute queries. Brackets can also be used to indicate in which order of precedence the query should be evaluated:

```
query {and|or} {not} (query)
```

For example:

```
{gsmcell} iid = “TestID” and bvid = “Test Project”
```

Note: This query expands on the first example by restricting the search to include a specific ENTERPRISE project.

Advanced EDS Query Syntax

In addition to the basic boolean expressions that the EDS query syntax supports (described in the table under Supported Operations), the following details the more advanced types of expression that may be required for certain data types.

"Not Equals" Query Operation Support

Queries where the value of an expression is not equal, for example:

```
iid neq "AB123"
{or}
iid != "AB123"
```

Will return all records that don't match AB123.

"Not" Expression Support

Negate an expression in EDS, for example

```
not iid = "AB123"
{or}
iid = "AB123" or not (iid = "CD456" or iid = "GH789")
```

"Contains" Expression Support

Search the contents of strings for a given expression, for example:

```
iid contains "GHJ"
```

Note: The substring will include the start of the string.

Regular Expression Report

EDS supports regular expression queries against string types. Internally these map onto the Oracle "REGEX_LIKE" function, a description of which can be found here:

http://docs.oracle.com/cd/B19306_01/appdev.102/b14251/adfns_regexp.htm#i1011021

For example:

```
iid regex "^AB(12|34)CDEF$"
```

EDS will search for either "AB12CDEF" or "AB34CDEF".

Subqueries

Subqueries allow EDS to return a parent type based on an expression applied to one if its children, only specific attributes support them listed in the table below:

Parent Type	Subquery Attribute	Child Type
Property		

Code updated to allow for new subquery syntax to exist on specific elements GSMCellv90 (subcells), CellSitev90 (cells), LTENodev90 (cells), NodeBv90Type (cells), AntennaDeviceType (patterns). New keyword allows parent node to be retrieved based on existence of child attribute. For example:

GSMCellv90 - exists {subcells : ModifyDate > \"2017-01-26\"} and bvid = "myproject"

NodeBv90Type - exists {cells : ModifyDate > \"2017-01-26\"} and bvid="myproject"

AntennaDeviceType - exists {patterns : ModifyDate > \"2017-01-26\"} and bvid="myproject"

Note: This feature should only be used when you know there are only a few affected elements otherwise database performance may struggle.

Supported Operations

This table lists the operations which can be performed against an attribute:

Operation	Description
=	Logical equals
eq	Logical equals
neq	Not equal
!=	Not equal
>	Greater than
gt	Greater than
>=	Greater than or equal
gte	Greater than or equal
<	Less than
<=	Less than or equal
lte	Less than or equal
ieq	Case sensitive equals
st	Starts with
end	End with
regex	Regular expression over string types
not	Negate a given expression
contains	Looks for a substring within a string
exists	Subquery expression – see Subqueries on page 64.

Supported Attributes

Chapter 11 contains a detail table of all queryable EDS attributes across the available data types.

About User Defined Fields

User defined fields (UDF) add a variation to the syntax of a freeform query. This is because the query needs to identify both the user defined field group as well as the value you would like to search for. The syntax for a user defined query is:

```
udf:["{group}"]<operation> <value>
```

The keyword udf: is used to tell the freeform query what it should expect as field group within the square brackets. The group name must be presented inside speech marks.

For example:

```
{gsmcell} udf:["Site Status"] = "On Air"
```

Where "Type" would be a Cell and the query is looking for a matching element whose Site Status field has been set to On Air.

User Defined Field Data Types

In general, the pick-list User Defined Field is the most frequently used, and is therefore the default search criterion used with the expression syntax above. If, however, you wish to query upon the other UDF types (integer, string, boolean, float, or date and time) you will need to indicate this in the query syntax.

The syntax to specify the data type within the query is as follows:

```
udf: {i, p, s, b, f, d} ["{group}"] <operation> <value>
```

Where:

- i = integer
- p = pick-list
- s = string
- b = boolean
- f = float
- d = date and time

You can choose one of the letters in the bracketed list applicable to the UDF type, so for example:

`udf: s ["group"] = "value"` - will do a string lookup

`udf: i ["group"] = 1234` - will do an integer lookup

`udf : d ["group"] = "yyyy-MM-dd HH:mm:ss zzz"` - will do a date and time lookup

Tip: In the last example above, "zzz" represents the offset from UTC (Universal Time Coordinated) in hours and minutes, with a leading + or -

6 Deploying and Configuring EDS

EDS Deployment Options

There are various ways in which EDS can be deployed, the choice of which is dependent on the host environment (such as single server or clustered) and the anticipated workload (such as targeted operations or bulk processing) that the API will support.

Here is a brief overview of the different deployment options:

- Setup A: Running EDS standalone as a single instance
- Setup B: Running EDS as a single server cluster using MSMQ
- Setup C: Running EDS as a single or multiple server cluster with RabbitMQ

Note: There are additional more advanced deployment options described under ASSET Event Publisher (AEP) on page 73.

Setup A: Running EDS Standalone as a Single Instance

EDS offers a single instance deployment option via the “Composite” runtime executable. It will allow multiple client requests to the API and serve them via its pool of (8) database worker threads. It therefore represents the simplest setup.

Strengths:

- Good for lightweight deployments, and it is easy to use and configure.
- Good for diagnostics and it is good for (API) development purposes.

Weaknesses:

- May be prone to single-point system failures, caused by application fault or server failure.

The EDS installation includes a binary named “TEOCO.EDS.Composite.Runtime”. This executable represents the single instance variant of EDS and its primary configuration settings are held in the file named “TEOCO.EDS.Composite.Runtime.config”.

Note: It is not possible to run the EDS composite at the same time as a clustered version of the EDS runtimes (specifically TEOCO.EDS.Presentation.Runtime.exe). If these are running interactively or as a Windows service the Composite may fail to start. Check the Windows Event Log for any issues that may occur on start up.

Setup B: Running EDS as a Single Server Cluster Using MSMQ

This configuration offers a single server deployment with multiple EDS components. This means that a single PRESENTATION service can talk to one or more (multiple) PERSISTENCE services via Microsoft message queue (MSMQ), which is a configurable option on Windows Server. Historically this has been the default deployment option that the EDS installer provided, and that is still the case.

It will deploy EDS in two parts:

1. A single-instance “Presentation” service, dedicated to serving SOAP and RESTful requests to its clients.
2. A multi-instance “Persistence” service that offers:
    - Built-In redundancy.
    - Increased robustness and availability to the presentation runtime.
    - Increased throughput to the database (dependent on Oracle server performance).
    - More resistance to single point of failures.

Note: You can in fact configure EDS over MSMQ to use multiple servers, rather than a single server, but such a configuration is beyond the scope of this document.

A clustered installation of EDS requires two configuration files for the Persistence and Presentation service components:

- TEOCO.EDS.Presentation.Runtime.config – This contains configuration pertinent to the front-end presentation service.
- TEOCO.EDS.Persistence.Runtime.config – This contains configuration regarding the backend persistence service.

Note: In addition to the primary configuration files used by the application, EDS will share common settings in a centralised set of config files to minimise repetition. See the next section for details.

Setup C: Running EDS as a Single or Multiple Server Cluster Using RabbitMQ

This is the most advanced deployment option for EDS. It is similar to deploying via MSMQ, but it uses an alternative and more scalable message broker (RabbitMQ) instead.

This high availability configuration offers several advantages over the standard MSMQ deployment:

- No longer necessary to confine an EDS cluster to single server.
- Support for multiple “Presentation” service instances that represent interaction with the client*.
- Built-In Redundancy.
- Increased throughput to the database (dependent on Oracle server performance).
- Greater resistance to single-point system failures.

Note: * If EDS is deployed with multiple servers with multiple presentation services, you may need to use a load balancer (such as Nginx or any suitable reverse proxy) to map a client connection to a number of presentation services. The configuration of using a load balancer is beyond the scope of this document.

See the following section for special requirements for this type of setup.

Enabling RabbitMQ Support

Given that the default deployment option that the EDS installer provides is for MSMQ, rather than RabbitMQ, you will need to modify the following configuration files:

- TEOCO.EDS.Persistence.Runtime.exe.config
- TEOCO.EDS.Presentation.Runtime.exe.config

In each file:

1. Uncomment the line containing the “MessageConfig” section for RabbitMQ:

```
<!-- For RabbitMQ switch to this config instead:
 <MessageConfig configSource="RabbitMQMessagePubSub.config" /> -->
```

2. Comment or remove the line containing the “MessageConfig” section for MSMQ.

```
<MessageConfig
configSource="MSMQMessagePubSub.config"></MessageConfig>
```

Ensure that you do this in both of the above-mentioned configuration files (Presentation and Persistence) and restart EDS.

For information on installing RabbitMQ, see the *EWS Installation Guide*.

Note: By default, the names of the queues are auto-created. If you want to have some control over this, you will need to amend the RabbitMQMessagePubSub.config file (in which AutoCreateQueues is set to "true").

Customising EDS Configuration

This section provides information on the significant sections within the EDSConfiguration.config file.

Environment Variable Substitution

Most of the values that you set in the config files can be replaced with a Windows environment variable, this can be very useful for passwords or other environment settings that you do not need to repeat manually.

In order to register an environment variable with EDS, define the variable as a “system wide” setting on Windows, then within the appropriate config file, substitute a hardcoded string with an environment variable as shown here:

Before:

```
    <database-login-configuration name="EDSDB" username="EDS_SOA_Client"
password="password" />
```

After:

```
    <database-login-configuration name="%EDSDB%"
username="EDS_SOA_Client"
password="%EDSPW%" />
```

In this example the DB connection name and the password have been replaced by environment variables named “EDSDB” and “EDSPW” respectively.

Common EDS Configuration Settings

The "EDSConfiguration.config" file contains a shared set of settings that can be used by the clustered or standalone runtime modes of EDS.

This table lists the most important sections within the EDSConfiguration.config file:

Section	Description	See
DTOTraceConfig*	Configuring EDS tracing options.	Configuring EDS Tracing Options on page 70
MultiTechNode*	MU-Node: create/update permissions.	Setting Permissions for Multi-Technology MU-Nodes on page 71
MultiTechCell*	Multiple-technology cell: create/update permisssions.	Setting Permissions for Multi-Technology Cells on page 71
DefaultSecurity	Part of this section relates to Impersonate User, which allows EDS to act as an 'ASSET Logged-in User'. It relates to Update, Create and Delete requests.	ASSET User Impersonation on page 72
EnterpriseEvent Logging	The ENTERPRISE event log stores committed data of all users so that a Refresh can show the network elements and any other objects in their currently Committed state. This section is used to switch event logging on ('true') or off ('false').	-

Note: Any changes to the settings in the sections of the file marked with an '*' (see the Section column in above table) are dynamic. That is, you do not have to shut down EDS and start it up again in order for the changes to take effect.

Configuring EDS Tracing Options

As part of its operation, EDS receives "Request" messages that are transmitted by clients and then replies to the client with a "Response" message.

The settings in the DTOTraceConfig section enable you to configure EDS to start dumping this data to disk, including the ability to separate out the embedded payload AppData for subsequent usage with EDS loader. This option is useful when troubleshooting faults with EDS however caution should be used with enabling the setting as it may consume a large amount of disk space.

Item	Description	Notes
Enabled	Sets the tracing on ('true') or off ('false').	
MinDiskSpaceMB	Minimum amount of disk space you want to set aside for tracing. EDS checks that space is available at the specified location.	
DumpFolder	Specified location for the tracing data. This location must be accessible to EDS.	

FilterItem	Select the type of data you want EDS to start dumping:
	
	Request = the data that EDS receives
	
	Response = the data that EDS returns
	
	AppData = the embedded payload data in the response (this data is also duplicated in the response message)
		Multiple items can be set, using comma-separation.

Item	Description	Notes
FilterErrorCodes	Select the error codes for which you want EDS to start dumping data: Failed = Only dump data when the entire message fails (Failed response code) Partial = Dump data when at least some of the content either has validation errors or other issues InvalidData = Dump data when at least some of the content has validation errors OK = Dump successful data	Multiple items can be set, using comma-separation.

You can see the default settings in this section of the EDSConfiguration.config file:

```
<DTOTraceConfig Enabled="false" MinDiskSpaceMB="50"
DumpFolder="%appdata%\EDS\Dump" FilterItem="Request, Response"
FilterErrorCodes="Failed" />
```

Setting Permissions for Multi-Technology MU-Nodes

Fixed MU-Nodes support only a single technology, whereas **Variable** MU-Nodes can support multiple configured technologies.

By default, the settings in the MultiTechNode section only allow EDS to create a **Fixed** MU-Node, and do not allow you to create a **Variable** MU-Node.

Also, by default, you cannot create any **Remote Cell Properties** or **Remote Cell Antennas**.

If you want to change any or all of these settings, you can edit this section of the config file according to your requirements.

You can see these default settings in this section of the EDSConfiguration.config file:

```
<MultiTechNode allowVariable="true" allowFixed="false"
 allowRemoteCellAntennas="false" allowRemoteCellProperties="false"/>
```

If required, ‘allowVariable’ and ‘allowFixed’ can both be set to ‘true’.

Setting Permissions for Multi-Technology Cells

By default, the settings in the MultiTechCell section only allow EDS to create or update a cell that supports a single technology. This means that it cannot create a cell that can support multiple configured technologies.

Also, the settings in the config file do not allow EDS to to change (update) the ‘Active’ technology on an existing cell.

If you want to change any of these settings, you can edit this section of the config file according to your requirements.

You can see the default settings in this section of the **EDSConfiguration.config** file:

```
<MultiTechCell
allowMultiTech="false"
allowActiveChange="false"/>
```

ASSET User Impersonation

This allows an EDS client to impersonate the credentials of a registered ASSET user. This feature may prove useful if you need to differentiate multiple EDS users who make changes to the ASSET database. The default behaviour of EDS is to write all changes under a special “EDS_SOA_Client” user registered within ASSET.

In the **EDSConfiguration.config** file you can switch on or switch off the ability to impersonate an ASSET User and set the Default User/Group that you want the EDS Service to impersonate. This user must exist as a genuine ASSET User and be a member of the specified Group.

If the specified Impersonation Group does not exist or the specified Impersonation User is not in the specified Impersonation Group, an error message is issued and the service shuts down.

The default state for Impersonate User is ‘False’ (off).

You can see the default settings in this section of the **EDSConfiguration.config** file:

```
<DefaultSecurity ... ImpersonationGroup="Administrators"
DefaultImpersonateUser="EDS_SOA_Client" AllowImpersonation="False"
```

When Impersonate User is turned ‘on’ you can also impersonate a user at the time of client request. This means that for each Update, Create and Delete request you send to EDS, you can also send a user name, and this will take precedence over the Impersonate User from the config file. Again, the user must be a valid ASSET User and a member of the Impersonation Group defined in the config file.

7 ASSET Event Publisher (AEP)

The AEP is a standalone EDS component that monitors activity within a configured ASSET database and broadcasts a simple event of change record to a RabbitMQ instance. Specifically, the AEP monitors ASSET User “commit” activity made against the “ENT_EVENTLOG_DATA” table within the ASSET database schema. This means that whenever an ASSET user commits an element from the Site Database or deletes an item from the Wastebasket for example, it can be tracked and broadcast to an external system. With this component, advanced integrations can be created with external solutions that await change notifications from the AEP and then may use EDS for enrichment.

Using the AEP API

Similar to EDS, the AEP includes a simple OpenAPI RESTful interface and also has a built in “Swagger” UI self-documenting Web Page. Unlike EDS however, this RESTful interface is only intended for administration and diagnostics, which will be described later in this section.

For reference, the AEP administration endpoint can be located (by default) on the following port:

```
http://{server}:8082
```

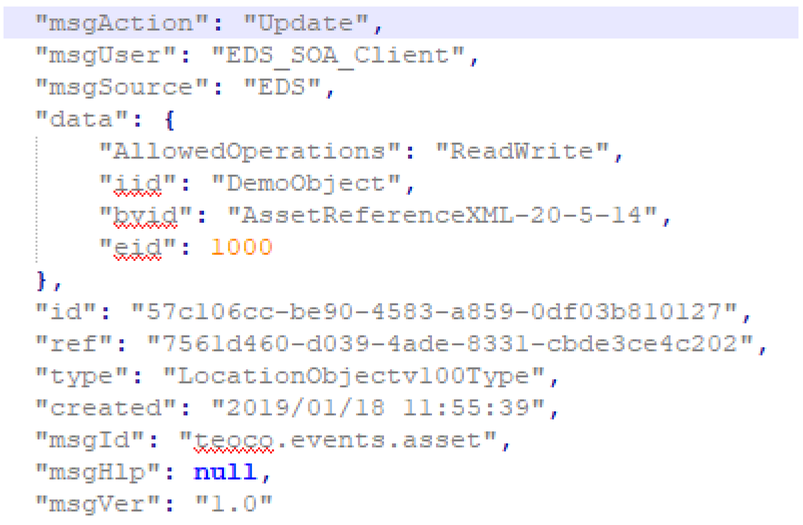
The primary function of the AEP is to broadcast events onto RabbitMQ, therefore normal operation is largely silent and the correct way to validate activity after initial deployment is to monitor the RabbitMQ administration console to confirm that events are being delivered to a specified RabbitMQ Queue.

AEP Event Structure

An AEP event is a simple JSON formatted canonical message that contains key details about an ASSET/EDS data type. The structure is deliberately sparse so as not to overwhelm a remote RabbitMQ message broker with a lot of “spam” from the ASSET database. It is assumed that once a systems integrator receiving AEP events builds their solution, they will use the key details from AEP to interrogate EDS for further detail – if required.

That said, the AEP nests a highly reduced version of an EDS data type within its message so some customisation is possible if consumers require more EDS data to be present.

The following example shows a typical AEP event:



```
  "msgAction": "Update",
  "msgUser": "EDS_SOA_Client",
  "msgSource": "EDS",
  "data": {
      "AllowedOperations": "ReadWrite",
      "iid": "DemoObject",
      "bvid": "AssetReferenceXML-20-5-14",
      "eid": 1000
  },
  "id": "57c106cc-be90-4583-a859-0df03b810127",
  "ref": "7561d460-d039-4ade-8331-cbde3ce4c202",
  "type": "LocationObjectv100Type",
  "created": "2019/01/18 11:55:39",
  "msgId": "teoco.events.asset",
  "msgHlp": null,
  "msgVer": "1.0"
```

This table describes the relevant fields from the AEP event:

Field	Description
msgAction	The type of AEP event, either "Update" or "Delete"
msgUser	The originating ASSET user ID who made the change
msgSource	The originating system that created the event, either EDS or ASSET.
data	The minimal EDS object containing specific data about the object affected. Typically this will be iid, bvid and CustomFields from the source. Note: this can be customised by manipulating the AEP Python scripts.
id	A unique message ID for the event
ref	The EDS reference ID used to enrich the event
type	The EDS datatype reflected by data
created	The timestamp of the event
msgId	A customisable string used to recognise AEP events
msgHlp	A customisable string used to indicate where online help might be located.
msgVer	A customisable string for versioning events

Note: The customisable strings are defined inside the "event-config" section inside the main TEOCO.EDS.Eventing.runtime.config. The AEP Python scripts are stored inside the "Plugins" folder that is part of AEP's deployment. A description of how to extend them is beyond the scope of this document.

Deploying and Configuring AEP

AEP integrates with ASSET and EDS using two different mechanisms in order to subscribe to commit events:

1. An Oracle trigger monitors the ASSET “ENT_EVENTLOGDATA” table for ASSET commit records and converts these into messages that are transmitted using Oracle Advanced Queuing (AQ). AEP uses a special database connection to subscribe to AW messages and convert them into its own.

2. A dedicated RabbitMQ subscription to listen to activity from EDS itself. This integration is necessary when EDS is used for updating the ASSET DB directly. EDS contains a dedicated config file to enable event forwarding to the AEP.

Using EDS for AEP Event Enrichment

AEP internally uses EDS to enrich any events received from ASSET therefore it is possible to either embed an EDS instance within the AEP environment, or connect AEP to an external EDS deployment using RabbitMQ as a transport. This behaviour is similar to the lightweight deployment option of using the EDS “Composite” component over using the more complete Presentation/Persistence deployment option. The former choice is convenient for small deployments or debugging, however the latter is more suitable to clustered enterprise scale integrations.

Note: Running an embedded instance of EDS within AEP will still consume an EDS “Persistence” license – please ensure you have sufficient licenses to use either configuration or contact TEOCO support for assistance.

AEP Configuration

The primary configuration file for AEP is named "TEOCO.EDS.Eventing.Runtime.config". Its composition is similar to EDS. This section describes the main parts of the AEP configuration that you may wish to alter:

Section	Description
MessageConfig	Used for connecting AEP to an external EDS instance. Designates whether MSMQ (default) or RabbitMQ should be used.
web-api	Configures the RESTful API port and hostname as well as capturing the API-Keys necessary for access control to the AEP administration features.
event-config	Primary configuration section for the AEP, see below.
database-login-configuration	Identical to that of EDS and is used to specify the database connection details and password. May also be encrypted using a command line or replaced with an environment variable.

Altering the "Event-Config" Section

This section contains several important configuration settings you may wish to alter within AEP:

Section	Default	Description
msg-id	teoco.events.asset	Used to recognise AEP events
msgver	1.0	Provides simple version control management
msghlp		Customisable help string
useEDSProxy	True	Configures AEP to either connect to an external EDS instance or embed it internally
WorkFlowThreads	4	The number of listening threads that await messages and perform enrichment against EDS

Specifying the RabbitMQ Connection Details

AEP and EDS need to connect to a pre-existing RabbitMQ broker. You can either specify this manually via each of the “Broker” attributes within the relevant config files, or you may find it more convenient to specify it via an environment variable named: EWS_BROKER_HOSTNAME.

Note: This environment variable can simply contain a fully qualified hostname, such as: “myserver.company.org”. However, if required, it can incorporate login credentials, for example: “<username>:<password>@myserver.company.org”.

By default, AEP will create two demo queues upon RabbitMQ named “AEPDemo1” and “AEPDemo2”. You can modify these via the config file within the “event-config” section add or remove them as required. For each “publisher” entry within the AEP config, it will send a duplicate message to RabbitMQ:

```
<publishers>
                  <rabbit name="AEP1" Provider="amqp:"
RequestQueue="topic://AEPEvents/AEPDemo1" Broker="localhost"
AutoCreateQueues="true" TimeToLive="150" RoutingKey="queue1" />
                  <rabbit name="AEP2" Provider="amqp:"
RequestQueue="topic://AEPEvents/AEPDemo2" Broker="localhost"
AutoCreateQueues="true" TimeToLive="150" RoutingKey="queue2" />
            </publishers>
```

Forwarding EDS Events to AEP

By default, EDS will not forward any commit or delete events to AEP without specifically altering a setting. Locate the “EventingPublisher.config” file within the deployment folder and change the “enabled” setting to “true”. You will need to restart EDS following this change.

Database Installation

For AEP to be usable, it must establish several tables within the ASSET database schema and grant itself appropriate permissions to use Oracle Advanced Queuing. The command line for DB installation is similar to that of EDS:

```
TEOCO.EDS.Eventing.Runtime.exe -installdb user=<admin-user> password
=<admin-pw> sid=<sid> [sysdba]
```

Where admin-user and password are either the ASSET administrator login or a DBA user account allowed to alter the schema.

The “sysdba” parameter is necessary for the first-time installation of AEP in order to grant the following:

```
grant execute on sys.dbms_aqadm to ent_schema_creation_role;
grant execute on sys.dbms_aq to ent_schema_creation_role;
```

Subsequent installations or upgrades of AEP may not require the sysdba option unless explicitly indicated in any release notes.

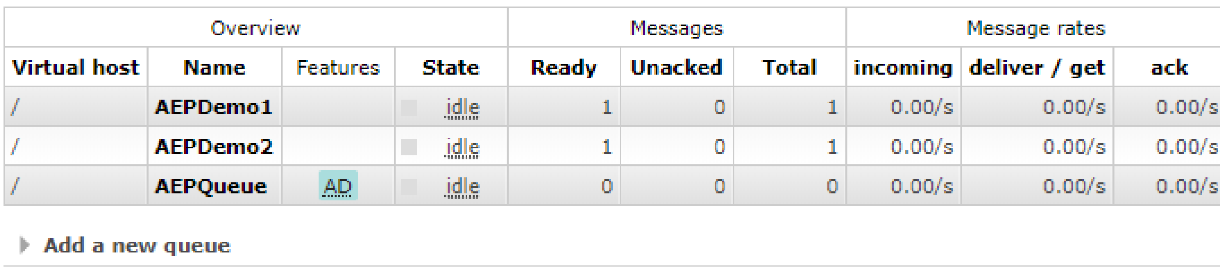
Note: An Oracle account with DBA permissions is required for AEP setup however if your DBA does not permit access to an elevated login, they can run the grants indicated above independently to the AEP installation and then a normal non-elevated DB installation omitting the “sysdba” parameter can be performed successfully.

Testing AEP

Once AEP has been successfully deployed and integrated with ASSET, EDS and RabbitMQ and you have confirmed that the service can be started without error, there are several methods you can use to validate correct operation:

1. Make a change to ASSET by committing something in the Site Database (for example).
2. Make a change to an item in EDS.
3. Use the AEP Management UI to generate a test message.

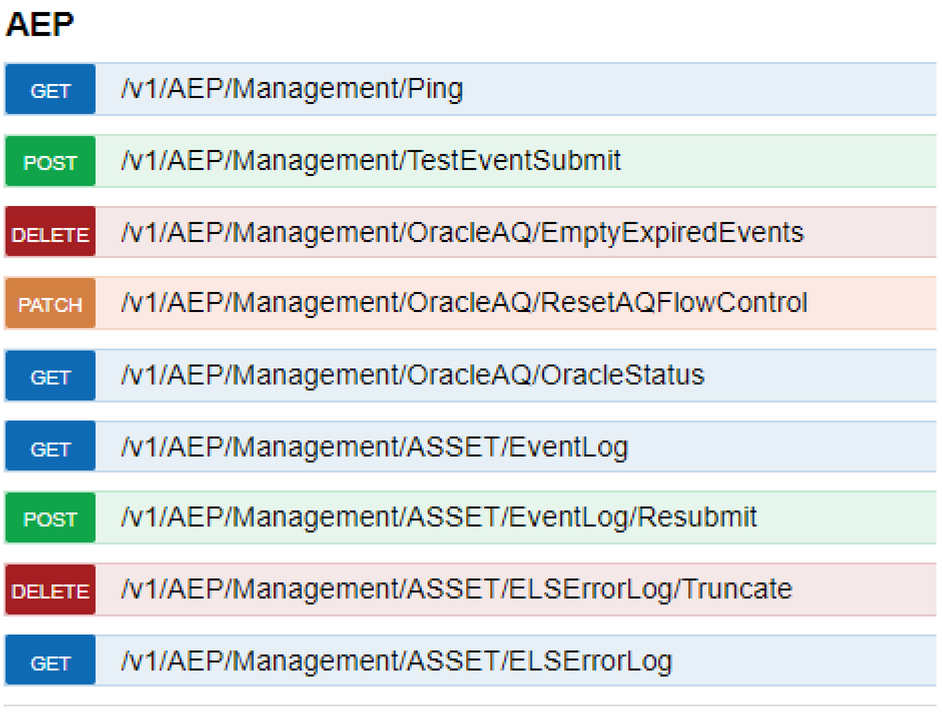
In all instances you should be able to see a resulting event generated and recorded via the RabbitMQ management console:



Overview				Messages			Message rates		
Virtual host	Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
/	AEPDemo1		idle	1	0	1	0.00/s	0.00/s	0.00/s
/	AEPDemo2		idle	1	0	1	0.00/s	0.00/s	0.00/s
/	AEPQueue	AD	idle	0	0	0	0.00/s	0.00/s	0.00/s

Using the AEP management RESTful API

AEP includes its own self documenting Swagger endpoint for diagnostic and management/monitoring activity, it can be accessed on `http://{server}:8082`


AEP

GET /v1/AEP/Management/Ping
POST /v1/AEP/Management/TestEventSubmit
DELETE /v1/AEP/Management/OracleAQ/EmptyExpiredEvents
PATCH /v1/AEP/Management/OracleAQ/ResetAQFlowControl
GET /v1/AEP/Management/OracleAQ/OracleStatus
GET /v1/AEP/Management/ASSET/EventLog
POST /v1/AEP/Management/ASSET/EventLog/Resubmit
DELETE /v1/AEP/Management/ASSET/ELSErrorLog/Truncate
GET /v1/AEP/Management/ASSET/ELSErrorLog

AEP API-Key Usage

Similar to EDS, usage of most of the API functions requires an API key to be registered and allocated to allow clients access to the method. There are two roles within AEP named “AEP” and “AEPManagement” and by default both can be granted via a default key named “aep”. This can be entered either directly via the SwaggerUI or as an HTTP header named “api-key”.

You can define or change the default api-key inside the main config file for AEP:

```
<web-api port="8082">
                ...
                <api-key key="aep">
                        <role name="AEP" />
                        <role name="AEPManagement" />
                </api-key>
        </web-api>
```

Note: As AEP can embed an instance of EDS within itself, you will also see API-key definition for EDS roles in the same section.

AEP API Function Descriptions

The following provides a short description of each of the available API functions within AEP:

```
GET /v1/AEP/Management/ASSET/ELSErrorLog
```

This will return the most recent error messages logged by EDS or AEP to the ASSET database. There are supporting parameters to influence the page size or the starting row.

```
DELETE /v1/AEP/Management/ASSET/ELSErrorLog/Truncate
```

This will permanently delete all data from the EDS/AEP error log table.

```
GET /v1/AEP/Management/OracleAQ/OracleStatus
```

This will indicate the Oracle AQ status, it can be used to indicate whether there is an issue with Oracle AW.

```
POST /v1/AEP/Management/TestEventSubmit
```

This will dump a simple test message onto AEP, which it will forward to RabbitMQ. You should see the result in the RabbitMQ console.

```
GET /v1/AEP/Management/ASSET/EventLog
```

This will list the current set of ASSET commit events from the ASSET database. It is useful for monitoring ASSET user activity that has occurred within the system. If advantageous, you can filter the activity on specific projects, data types or users.

```
POST /v1/AEP/Management/ASSET/EventLog/Resubmit
```

Essentially the same method as the previous one (EventLog), however this can be used to “replay” activity from the ASSET event log onto AEP. It may prove useful when debugging, but it is not intended as a primary means of driving the component.

8 EDS Fault Diagnosis

Diagnosing Faults within EDS

The EDS is preconfigured to log any serious fault it encounters to three locations, namely:

- To the client in the form of a Status message. The textual element is used to indicate where and why the fault occurred.
- To the Windows Event Log on the server hosting the EDS.
- To the ENTERPRISE database, where the Administrator can see the faults using the dedicated web content written to monitor the EDS.

In addition, the EDS uses the Microsoft Enterprise Library Logging Block for filtering and forwarding faults or other diagnostic information onto a number of subscribed listeners. It is possible to add new listeners to the EDS configuration through modification of the application's .config file. In this way, it would be possible to include a listener which forwards serious faults to a specified e-mail address.

Note: The EDS application bundle includes a Microsoft Enterprise Library config editor to allow administrators to alter the configuration of these logs. For more information, see the *EWS Web Administration and User Guide*.

If the EDS raises an unhandled fault or exception, its services will attempt to continue normal operation. Most faults or exceptions are based around validation failure, or at the worst a rejection from the Oracle server due to, for example, constraint violation. In all of these circumstances, the EDS will log the failure and continue normal operation.

Manually Performing the EDS Database Setup

It is possible to use a command-line process to install EDS in a new target database, or reinstall and/or upgrade an existing database. This may be relevant when a patch to EDS has caused its internal database version to be updated. The command line option may be more convenient, especially when attempting to troubleshoot a failed installation or when the services do not start.

To use the database installation command line, type the following (from the EDS application folder):

```
teoco.eds.persistence.runtime.exe -installdb user=<user>
password=<password> sid=<oracle_service>
```

Where `<user>`, `<password>` and `<oracle_service>` may be substituted by the appropriate values.

Notes:

- If you want to perform an EDS database installation, you must supply either the ENTERPRISE administrator account or another user ID capable of updating the database schema (NETWORK_PLANNING)
- If the details you have entered are correct, EDS will begin updating the database and present the changes it makes to the console as well as appending them to the "installdb.log" file

---

Manually Changing the EDS Database Connection String

By default, when EDS is installed it stores an encrypted connection string inside multiple configuration files containing the EDS_SOA_Client password and Oracle service name.

Note: This connection string is present in the following files:

- TEOCO.EDS.Persistence.Runtime.exe.config
- TEOCO.EDS.Presentation.Runtime.exe.config
- TEOCO.ENS.Runtime.exe.config

To change the connection string manually:

1. Back up each of the config files before performing this process. If this is not done correctly, the services will not start.
2. Open one of the config files and locate the xml section named “database-login-configuration”. The encrypted section will look like this example:

```
<database-login-configuration
configProtectionProvider="DataProtectionConfigurationProvider">

    <EncryptedData>

        <CipherData>
<CipherValue>AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAZICI6NQKaEy0QPHVA8+Ubg
QAAAACAAAAAAAADZgAAwAAAABAAAADlyoeQ6pOkoVhext6s/DJBAAAAAASAAACgAAAAE
AAAAG03swS/6fPPBRqeFeeT0wm4AAAABNJBm1UADlCuqJ5THBuHn+IJKsTdEp/p7JwJ
CBF8s3w8wKvPHKwPR4jly8fcvaWcaCtzpbUd3IkCCMhT4RhsWsooTQQ6MDdyXKeJwRq
5jxNaN0jU5AaRt7zHn7LhSgip4rrTHjzzptRiCrhjYuJGX7Ljra2qnwwZbzEGMg48tA
MAdf0heg+Mw+S08WjbMP/4VGFmcmwUGRJ7hhxSX9TfAwT6AoYf8CWqRDS4SdG4u+b6L
tyMZxR52hQAAAAuf1cTv5fNJDAMwJK6LTKn3hpjRQ==</CipherValue>

        </CipherData>

    </EncryptedData>

   </database-login-configuration>
```

3. Having located the correct section, replace the entire xml element with the following XML.

   Note: Substitute “EDSDB” and “password” with the appropriate Oracle SID and password for the “EDS_SOA_Client” account.

   <database-login-configuration name="EDSDB" username="EDS_SOA_Client" password="password">

   </database-login-configuration>

4. Perform this process for each of the 3 config files.

   Note: To test whether the change has worked, either restart the services from the service control panel or interactively using the command shell.

5. To re-encrypt the connection string, type the following from the command line:

```
TEOCO.EDS.Persistence.Runtime.exe -encrypt database-login-
configuration
```

```
TEOCO.EDS.Presentation.Runtime.exe -encrypt database-login-
configuration

TEOCO.ENS.Runtime.exe -encrypt database-login-configuration
```

Performance Monitoring

The EDS records several counters and other values for describing message throughput, system activity and numbers of successful and unsuccessful messages it has processed. This table lists these values:

Name	Area	Description
Machine Name	General	The name of the server hosting the EDS.
Process Name	General	The name of the server process.
Process ID	General	The server process ID.
NonpagedSystemMemory	General	The amount of non-paged system memory.
PagedMemorySize	General	The amount of paged memory.
PeakPagedMemorySize	General	The peak amount of paged memory.
PeakVirtualMemorySize	General	The peak amount of virtual memory.
PeakWorkingSet	General	The working set size of the process.
InstanceName	General	Indicates whether the EDS is running as a stand-alone program or as a windows cluster.
NumberOfMessages Received	Create, Update, Query, Delete	Indicates how many messages the service has received.
NumberOfSuccessMessages Sent	Create, Update, Query, Delete	The number of success messages sent.
NumberOfFailureMessages Sent	Create, Update, Query, Delete	The number of failure messages. These could be for failure of validation criteria, for example.
NumberOfExceptionMessages Sent	Create, Update, Query, Delete	Number of serious errors, which should be monitored, as they might reflect a more serious issue with the system.
MinimumTimeForProcess Message	Create, Update, Query, Delete	The minimum amount of time the EDS took to handle a request.
AverageTimeForProcessing Message	Create, Update, Query, Delete	The longest time it too the EDS to answer a request.

Accessing Performance Values

The EDS offers four mechanisms for reading the values mentioned in the above table in the Performance Monitoring topic. They are:

- Using the REST interface
- Using the Web Service Interface
- Using Windows Performance Monitor
- Using the Web Administration Content

Accessing Performance Values with the REST Interface

The REST interface has an additional method which allows you to retrieve a raw XML dump of values. This URL can be used to access this:

```
http://{host} : {port}/Aircom/EDS/REST/status/
```

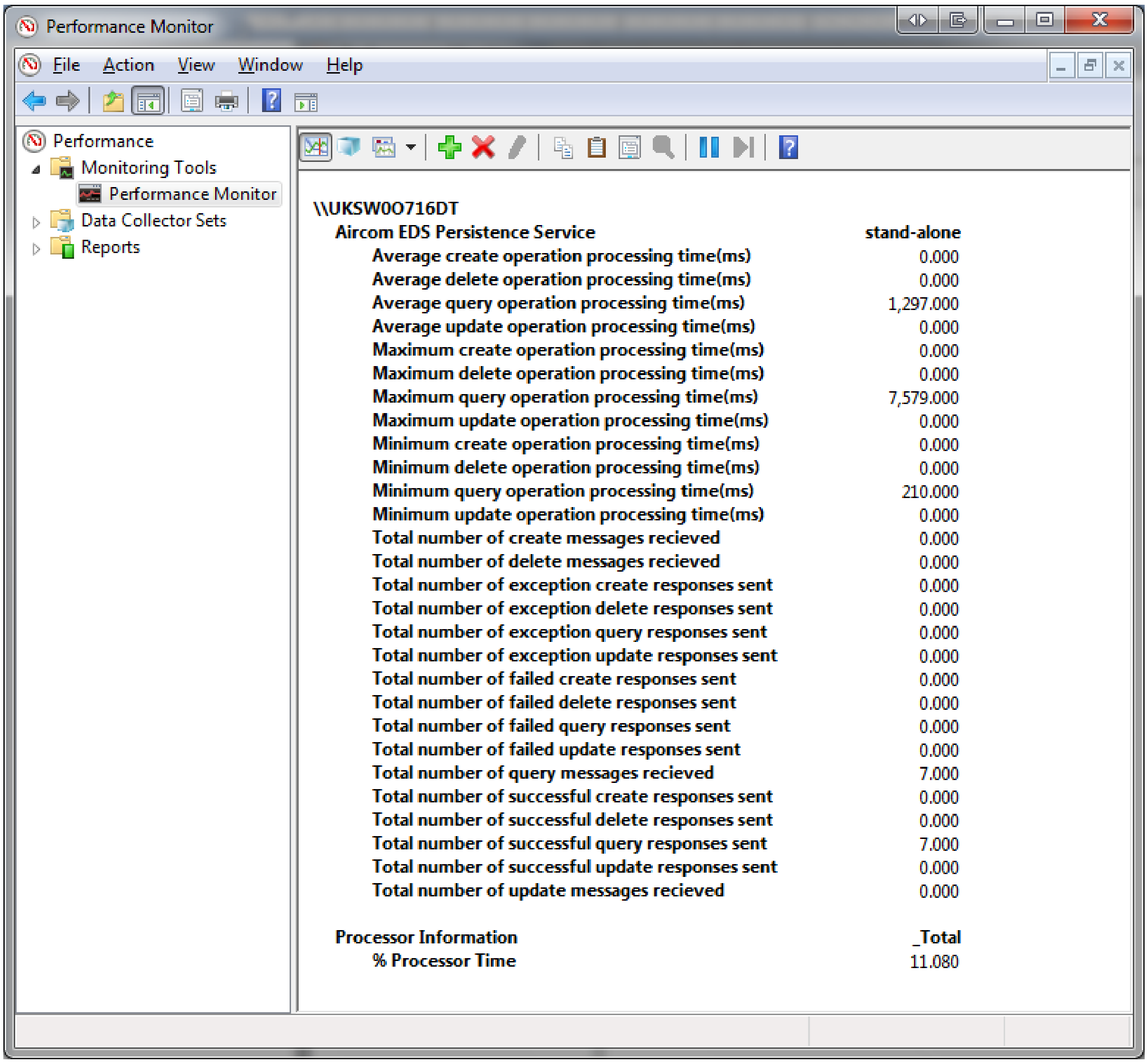
In response, the EDS will return a report similar to this example:

```
<SystemStatus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <Status>
            <Item xsi:type="EDSHeartBeatPacket" HostID="46e136c2-4c3f-
4d11-8282-10b739a2cf3f">
                  <ResponseTime>2010-03-
18T10:32:23.016764+00:00</ResponseTime>
                  <MachineName>UKSW0O716DT</MachineName>

      <ProcessName>Aircom.EDS.Persistence.Runtime</ProcessName>
                  <ProcessID>4612</ProcessID>

      <NonpagedSystemMemorySize>25612</NonpagedSystemMemorySize>
                  <PagedMemorySize>104464384</PagedMemorySize>
                  <PagedSystemMemorySize>553916</PagedSystemMemorySize>
                  <PeakPagedMemorySize>158646272</PeakPagedMemorySize>

      <PeakVirtualMemorySize>371535872</PeakVirtualMemorySize>
                  <PeakWorkingSet>107069440</PeakWorkingSet>
                  <StartTime>2010-03-
18T10:32:13.6886316+00:00</StartTime>
                  <VirtualMemorySize>371470336</VirtualMemorySize>
                  <WorkingSet>107057152</WorkingSet>
                  <InstanceName>Stand-Alone</InstanceName>
                  <Creates>

      <NumberOfMessagesRecieved>0</NumberOfMessagesRecieved>

      <NumberOfSuccessMessagesSent>0</NumberOfSuccessMessagesSent>

      <NumberOfFailureMessagesSent>0</NumberOfFailureMessagesSent>

      <NumberOfExceptionMessagesSent>0</NumberOfExceptionMessagesSent>

      <MinimumTimeForProcessedMessage>0</MinimumTimeForProcessedMessage>

      <AverageTimeForProcessingMessage>0</AverageTimeForProcessingMessage
>

      <MaximumTimeForProcessedMessage>0</MaximumTimeForProcessedMessage>
                  </Creates>
                  <Reads>

      <NumberOfMessagesRecieved>0</NumberOfMessagesRecieved>

      <NumberOfSuccessMessagesSent>0</NumberOfSuccessMessagesSent>

      <NumberOfFailureMessagesSent>0</NumberOfFailureMessagesSent>

      <NumberOfExceptionMessagesSent>0</NumberOfExceptionMessagesSent>

      <MinimumTimeForProcessedMessage>0</MinimumTimeForProcessedMessage>
      <AverageTimeForProcessingMessage>0</AverageTimeForProcessingMessage
>

      <MaximumTimeForProcessedMessage>0</MaximumTimeForProcessedMessage>
            </Reads>
            <Updates>

      <NumberOfMessagesRecieved>0</NumberOfMessagesRecieved>

      <NumberOfSuccessMessagesSent>0</NumberOfSuccessMessagesSent>

      <NumberOfFailureMessagesSent>0</NumberOfFailureMessagesSent>

      <NumberOfExceptionMessagesSent>0</NumberOfExceptionMessagesSent>

      <MinimumTimeForProcessedMessage>0</MinimumTimeForProcessedMessage>

      <AverageTimeForProcessingMessage>0</AverageTimeForProcessingMessage
>

      <MaximumTimeForProcessedMessage>0</MaximumTimeForProcessedMessage>
            </Updates>
            <Deletes>

      <NumberOfMessagesRecieved>0</NumberOfMessagesRecieved>

      <NumberOfSuccessMessagesSent>0</NumberOfSuccessMessagesSent>

      <NumberOfFailureMessagesSent>0</NumberOfFailureMessagesSent>

      <NumberOfExceptionMessagesSent>0</NumberOfExceptionMessagesSent>

      <MinimumTimeForProcessedMessage>0</MinimumTimeForProcessedMessage>

      <AverageTimeForProcessingMessage>0</AverageTimeForProcessingMessage
>

      <MaximumTimeForProcessedMessage>0</MaximumTimeForProcessedMessage>
            </Deletes>
         </Item>
      </Status>
</SystemStatus>
```

Note: There will be one "item" per EDS persistent service instance.

Accessing Performance Values with Windows Performance Monitor

Windows "Perfmon" can also be used to connect to the EDS persistent service. This picture shows the counters:



\\UKSW00716DT

Aircom EDS Persistence Service	**stand-alone**
Average create operation processing time(ms)	0.000
Average delete operation processing time(ms)	0.000
Average query operation processing time(ms)	1,297.000
Average update operation processing time(ms)	0.000
Maximum create operation processing time(ms)	0.000
Maximum delete operation processing time(ms)	0.000
Maximum query operation processing time(ms)	7,579.000
Maximum update operation processing time(ms)	0.000
Minimum create operation processing time(ms)	0.000
Minimum delete operation processing time(ms)	0.000
Minimum query operation processing time(ms)	210.000
Minimum update operation processing time(ms)	0.000
Total number of create messages recieved	0.000
Total number of delete messages recieved	0.000
Total number of exception create responses sent	0.000
Total number of exception delete responses sent	0.000
Total number of exception query responses sent	0.000
Total number of exception update responses sent	0.000
Total number of failed create responses sent	0.000
Total number of failed delete responses sent	0.000
Total number of failed query responses sent	0.000
Total number of failed update responses sent	0.000
Total number of query messages recieved	7.000
Total number of successful create responses sent	0.000
Total number of successful delete responses sent	0.000
Total number of successful query responses sent	7.000
Total number of successful update responses sent	0.000
Total number of update messages recieved	0.000
Processor Information	**_Total**
% Processor Time	11.080

Note: All time intervals displayed are in milliseconds.

9 EDS Request Response Samples

Creating Data within the EDS

A create request resembles the following XML example:

```
<Create xmlns="http://www.aircominternational.com/contract/EDS/2009/05">
      <data>
            <itemID
xmlns="http://www.aircominternational.com/contract/Util/2009/10">88384db3
-e9dc-411b-b34a-27f389bc79fd</itemID>
            <TimeoutOverride
xmlns="http://www.aircominternational.com/contract/EDS/DST/2009/10">0</Ti
meoutOverride>
                  <RunSequentially
xmlns="http://www.aircominternational.com/contract/EDS/DST/2009/10">false
</RunSequentially>
            <CreateItems a:objectType="NodeBType"
xmlns:a="http://www.aircominternational.com/contract/EDS/DST/2009/10">
                  <itemID
xmlns="http://www.aircominternational.com/contract/Util/2009/10">7d297283
-f9b9-497e-8299-593439da9a40</itemID>
                  <NewData>
                        <AppData xsi:type="q1:NodeBType" b:iid="NodeBA"
b:bvid="Northeast"
xmlns:b="http://www.aircominternational.com/Schemas/CommonTypes/2009/05"
xmlns:q1="http://www.aircominternational.com/Schemas/UMTS/2009/05">
                              <Location b:iid="4DET391B"
xmlns=""></Location>
                              <q1:NodeBID>-1</q1:NodeBID>
                              <q1:NodeBEquipmentType b:iid="Node B with
TMA"></q1:NodeBEquipmentType>
                              <q1:PLMN b:iid="PLMN0"></q1:PLMN>

      <q1:AllowedOperations>ReadWrite</q1:AllowedOperations>
                        </AppData>
                        <AppData xsi:type="q2:NodeBType" b:iid="NodeBB"
b:bvid="Northeast"
xmlns:b="http://www.aircominternational.com/Schemas/CommonTypes/2009/05"
xmlns:q2="http://www.aircominternational.com/Schemas/UMTS/2009/05">
                              <Location b:iid="4DET391B"
xmlns=""></Location>
                              <q2:NodeBID>-1</q2:NodeBID>
                              <q2:NodeBEquipmentType b:iid="Node B with
TMA"></q2:NodeBEquipmentType>
                              <q2:PLMN b:iid="PLMN0"></q2:PLMN>

      <q2:AllowedOperations>ReadWrite</q2:AllowedOperations>
                        </AppData>
                        <AppData xsi:type="q3:NodeBType" b:iid="NodeBC"
b:bvid="Northeast"
xmlns:b="http://www.aircominternational.com/Schemas/CommonTypes/2009/05"
xmlns:q3="http://www.aircominternational.com/Schemas/UMTS/2009/05">
                              <Location b:iid="4DET391B"
xmlns=""></Location>
                              <q3:NodeBID>-1</q3:NodeBID>
                              <q3:NodeBEquipmentType b:iid="Node B with
TMA"></q3:NodeBEquipmentType>
                              <q3:PLMN b:iid="PLMN0"></q3:PLMN>
      <q3:AllowedOperations>ReadWrite</q3:AllowedOperations>
                        </AppData>
                  </NewData>
            </CreateItems>
      </data>
</Create>
```

The example shows a simple request which creates three minimal NodeB types. The request has defined an itemID for both the other RequestType and the contained RequestItem. The response from the EDS looks like this:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
      <s:Header>
            <Action s:mustUnderstand="1"
xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">http://ww
w.aircominternational.com/contract/EDS/2009/05/EDS/CreateResponse</Action
>
      </s:Header>
      <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <CreateResponse
xmlns="http://www.aircominternational.com/contract/EDS/2009/05">
                  <CreateResult d4p1:itemIDRef="88384db3-e9dc-411b-b34a-
27f389bc79fd" timeStamp="2010-03-17T19:37:55.0813608+00:00"
xmlns:d4p1="http://www.aircominternational.com/contract/Util/2009/10">
                        <d4p1:Status code="OK" ref="88384db3-e9dc-411b-
b34a-27f389bc79fd">
                              <d4p1:Status code="OK" ref="7d297283-f9b9-
497e-8299-593439da9a40" />
                        </d4p1:Status>
                  </CreateResult>
            </CreateResponse>
      </s:Body>
</s:Envelope>
```

Note: If you were to create the same three NodeBs again, the EDS would return an error, shown in this example:

```
<CreateResponse
xmlns="http://www.aircominternational.com/contract/EDS/2009/05">
      <CreateResult d4p1:itemIDRef="e9821e0f-5b2a-43cd-841f-64849b578ef9"
timeStamp="2010-03-17T19:41:26.705465+00:00"
xmlns:d4p1="http://www.aircominternational.com/contract/Util/2009/10">
            <d4p1:Status code="Failed" ref="e9821e0f-5b2a-43cd-841f-
64849b578ef9">
                  <d4p1:Status code="InvalidData" ref="eff48a14-224a-
4fcd-ac30-2b8a1aab6e5c" comment="Object ID:NodeBA, Key:1018 has failed
validation : ObjectName 'NodeBA' is not unique (ObjectName must be unique
within the DB), 'ObjectName': NodeBAObject ID:NodeBB, Key:1019 has failed
validation : ObjectName 'NodeBB' is not unique (ObjectName must be unique
within the DB), 'ObjectName': NodeBBObject ID:NodeBC, Key:1020 has failed
validation : ObjectName 'NodeBC' is not unique (ObjectName must be unique
within the DB), 'ObjectName': NodeBC" />
            </d4p1:Status>
      </CreateResult>
</CreateResponse>
```

Updating Data within the EDS

Modify requests follow a similar pattern to a Create requests, but they are required to supply a query to identify the item they must replace. The query must always return a single result only, otherwise the EDS will reject the modify attempt.

An example of an update to the same three NodeBs used in the previous example is:

```
<Modify xmlns="http://www.aircominternational.com/contract/EDS/2009/05">
      <data>
            <itemID
xmlns="http://www.aircominternational.com/contract/Util/2009/10">a1ae6454
-c251-455c-8e77-2c5ab3af2da2</itemID>
            <TimeoutOverride
xmlns="http://www.aircominternational.com/contract/EDS/DST/2009/10">0</Ti
meoutOverride>
            <RunSequentially
xmlns="http://www.aircominternational.com/contract/EDS/DST/2009/10">false
</RunSequentially>
            <ModifyItems a:objectType="NodeBType"
a:CreateIfNotFound="false"
xmlns:a="http://www.aircominternational.com/contract/EDS/2009/05">
                  <itemID
xmlns="http://www.aircominternational.com/contract/Util/2009/10">8c89c01a
-20bc-452f-89c2-de79478aafca</itemID>
                  <a:Select>
                        <OverrideCacheDefault UseQueryCache="true"
xmlns="http://www.aircominternational.com/Schemas/Query/2009/09"></Overri
deCacheDefault>
                        <Simple
xmlns="http://www.aircominternational.com/Schemas/Query/2009/09">
                              <Query>iid="NodeBA" and
bvid="Northeast"</Query>
                        </Simple>
                  </a:Select>
                  <a:NewData>
                        <a:AppData xsi:type="q1:NodeBType" b:iid="NodeBA"
b:bvid="Northeast"
xmlns:b="http://www.aircominternational.com/Schemas/CommonTypes/2009/05"
xmlns:q1="http://www.aircominternational.com/Schemas/UMTS/2009/05">
                              <Location b:iid="4DET391B"
xmlns=""></Location>
                              <q1:NodeBID>-1</q1:NodeBID>
                              <q1:NodeBEquipmentType b:iid="Node B with
TMA"></q1:NodeBEquipmentType>
                              <q1:PLMN b:iid="PLMN0"></q1:PLMN>

      <q1:AllowedOperations>ReadWrite</q1:AllowedOperations>
                        </a:AppData>
                  </a:NewData>
            </ModifyItems>
            <ModifyItems a:objectType="NodeBType"
a:CreateIfNotFound="false"
xmlns:a="http://www.aircominternational.com/contract/EDS/2009/05">
                  <itemID
xmlns="http://www.aircominternational.com/contract/Util/2009/10">7c9e1c57
-cf73-415b-9150-8ebc32147981</itemID>
                  <a:Select>
                        <OverrideCacheDefault UseQueryCache="true"
xmlns="http://www.aircominternational.com/Schemas/Query/2009/09"></Overri
deCacheDefault>
```

```
<Simple xmlns="http://www.aircominternational.com/Schemas/Query/2009/09">
                              <Query>iid="NodeBB" and
bvid="Northeast"</Query>
                        </Simple>
                  </a:Select>
                  <a:NewData>
                        <a:AppData xsi:type="q2:NodeBType" b:iid="NodeBB"
b:bvid="Northeast"
xmlns:b="http://www.aircominternational.com/Schemas/CommonTypes/2009/05"
xmlns:q2="http://www.aircominternational.com/Schemas/UMTS/2009/05">
                              <Location b:iid="4DET391B"
xmlns=""></Location>
                              <q2:NodeBID>-1</q2:NodeBID>
                              <q2:NodeBEquipmentType b:iid="Node B with
TMA"></q2:NodeBEquipmentType>
                              <q2:PLMN b:iid="PLMN0"></q2:PLMN>

      <q2:AllowedOperations>ReadWrite</q2:AllowedOperations>
                        </a:AppData>
                  </a:NewData>
            </ModifyItems>
            <ModifyItems a:objectType="NodeBType"
a:CreateIfNotFound="false"
xmlns:a="http://www.aircominternational.com/contract/EDS/2009/05">
                  <itemID
xmlns="http://www.aircominternational.com/contract/Util/2009/10">e9995793
-ce1e-4d30-9b24-3a0ee901dea9</itemID>
                  <a:Select>
                        <OverrideCacheDefault UseQueryCache="true"
xmlns="http://www.aircominternational.com/Schemas/Query/2009/09"></Overri
deCacheDefault>
                        <Simple
xmlns="http://www.aircominternational.com/Schemas/Query/2009/09">
                              <Query>iid="NodeBC" and
bvid="Northeast"</Query>
                        </Simple>
                  </a:Select>
                  <a:NewData>
                        <a:AppData xsi:type="q3:NodeBType" b:iid="NodeBC"
b:bvid="Northeast"
xmlns:b="http://www.aircominternational.com/Schemas/CommonTypes/2009/05"
xmlns:q3="http://www.aircominternational.com/Schemas/UMTS/2009/05">
                              <Location b:iid="4DET391B"
xmlns=""></Location>
                              <q3:NodeBID>-1</q3:NodeBID>
                              <q3:NodeBEquipmentType b:iid="Node B with
TMA"></q3:NodeBEquipmentType>
                              <q3:PLMN b:iid="PLMN0"></q3:PLMN>

      <q3:AllowedOperations>ReadWrite</q3:AllowedOperations>
                        </a:AppData>
                  </a:NewData>
            </ModifyItems>
      </data>
</Modify>
```

Note: In this example, multiple ModifyItems as a ModifyItem may only update a single item at a time based on the restrictions of the query, which is necessary to correlate to the query (Select element). The query is using the simple free-text notation.

The response from the EDS resembles this:

```
<ModifyResponse
xmlns="http://www.aircominternational.com/contract/EDS/2009/05">
      <ModifyResult d4p1:itemIDRef="a1ae6454-c251-455c-8e77-2c5ab3af2da2"
timeStamp="2010-03-17T19:45:35.4616931+00:00"
xmlns:d4p1="http://www.aircominternational.com/contract/Util/2009/10">
            <d4p1:Status code="OK" ref="a1ae6454-c251-455c-8e77-
2c5ab3af2da2">
                  <d4p1:Status code="OK" ref="8c89c01a-20bc-452f-89c2-
de79478aafca" />
                  <d4p1:Status code="OK" ref="7c9e1c57-cf73-415b-9150-
8ebc32147981" />
                  <d4p1:Status code="OK" ref="e9995793-ce1e-4d30-9b24-
3a0ee901dea9" />
            </d4p1:Status>
      </ModifyResult>
</ModifyResponse>
```

Deleting Data within the EDS

Using the same examples as before, we can delete one of the NodeBs created previously. The query expression looks like this:

```
<Delete xmlns="http://www.aircominternational.com/contract/EDS/2009/05">
      <data>
            <itemID
xmlns="http://www.aircominternational.com/contract/Util/2009/10">a7255006
-cfb8-4642-9ab6-458600252b94</itemID>
            <TimeoutOverride
xmlns="http://www.aircominternational.com/contract/EDS/DST/2009/10">0</Ti
meoutOverride>
            <RunSequentially
xmlns="http://www.aircominternational.com/contract/EDS/DST/2009/10">false
</RunSequentially>
            <DeleteItems>
                  <itemID
xmlns="http://www.aircominternational.com/contract/Util/2009/10">8468f38a
-8686-4260-b54b-81c0c519b826</itemID>
                  <objectType>NodeBType</objectType>
                  <Select>
                        <OverrideCacheDefault UseQueryCache="true"
xmlns="http://www.aircominternational.com/Schemas/Query/2009/09"></Overri
deCacheDefault>
                        <Simple
xmlns="http://www.aircominternational.com/Schemas/Query/2009/09">
                              <Query>iid ieq "nodeba" and
bvid="northeast"</Query>
                        </Simple>
                  </Select>
            </DeleteItems>
      </data>
</Delete>
```

Note: The query expression will select a single NodeB named "nodeba" using a case sensitive “ieq” match, and will delete it and its dependencies, if any.

The EDS response to this query expression is similar to this example:

```
<DeleteResponse
xmlns="http://www.aircominternational.com/contract/EDS/2009/05">
      <DeleteResult d4p1:itemIDRef="a7255006-cfb8-4642-9ab6-458600252b94"
xmlns:d4p1="http://www.aircominternational.com/contract/Util/2009/10">
            <d4p1:Status code="OK" ref="a7255006-cfb8-4642-9ab6-
458600252b94">
                  <d4p1:Status code="OK" ref="8468f38a-8686-4260-b54b-
81c0c519b826" />
            </d4p1:Status>
      </DeleteResult>
</DeleteResponse>
```

Querying Data within the EDS

Once the NodeB has been deleted, you can query the remaining NodeBs from the database. The query looks like this:

```
<Query
      <data>
            <itemID>92c14dd7-4ff5-4871-9633-278d4380891a</itemID>
            <TimeoutOverride>0</TimeoutOverride>
            <RunSequentially>false</RunSequentially>
            <QueryItems a:objectType="NodeBType" count="100" offset="0"
xmlns:a="">
                  <itemID>477f994e-4dc4-4cba-a44c-118d32769199</itemID>
                  <Select>
                        <OverrideCacheDefault
UseQueryCache="true"></OverrideCacheDefault>
                        <Simple>
                              <Query>iid st "nodeb" and
bvid="northeast"</Query>
                        </Simple>
                  </Select>
            </QueryItems>
      </data>
</Query>
```

The query expression is selecting all NodeBs whose name starts with (st) “NodeB”.

If you wish, you can further refine the search criteria to NodeB objects that have been modified within a date and time range.

The query looks like this:

```
<Query>
      <data>
            <itemID>92c14dd7-4ff5-4871-9633-278d4380891a</itemID>
            <TimeoutOverride xmlns="">0</TimeoutOverride>
            <RunSequentially>false</RunSequentially>
            <QueryItems a:objectType="NodeBType" count="100" offset="0"
xmlns:a="">
                  <itemID>477f994e-4dc4-4cba-a44c-118d32769199</itemID>
                  <Select>
                        <OverrideCacheDefault
UseQueryCache="true"></OverrideCacheDefault>
                        <Simple>
                              <Query>iid st "nodeb" and bvid="project1"
and modifydate gt "2010-03-17T19:37:50" and modifydate lt "2010-03-
17T19:37:59"</Query>
                        </Simple>
                  </Select>
            </QueryItems>
      </data>
</Query>
```

Note: Date time formats can be one of the following:

- Date: “14/07/2010” or “2010-07-14” (dd/mm/yyyy or yyyy-mm-dd).
- Date and Time: “14/07/2010 12:00:01” or “2010-07-14T12:00:01” (dd/mm/yyyy hh:mm:ss or yyyy-mm-ddThh:mm:ss). Minutes and seconds are optional.

The result from EDS looks similar to this example:

```
<QueryResponse
xmlns="http://www.aircominternational.com/contract/EDS/2009/05">
      <QueryResult itemIDRef="92c14dd7-4ff5-4871-9633-278d4380891a"
timeStamp="2010-03-17T20:04:28.2494848+00:00"
xmlns:d4p1="http://www.aircominternational.com/contract/Util/2009/10">
            <Status code="OK" ref="92c14dd7-4ff5-4871-9633-278d4380891a">
                  <Status code="OK" ref="477f994e-4dc4-4cba-a44c-
118d32769199" />
            </Status>
            <Data d4p1:itemIDRef="477f994e-4dc4-4cba-a44c-118d32769199"
remaining="0" nextOffset="0">
                  <AppData
xmlns:q5="http://www.aircominternational.com/Schemas/UMTS/2009/05"
xsi:type="q5:NodeBType" d6p1:iid="NodeBB" d6p1:bvid=" project1"
d6p1:eid="1016"
xmlns:d6p1="http://www.aircominternational.com/Schemas/CommonTypes/2009/0
5">
                        <Security xmlns="">
                              <CreateDate>2010-03-
17T19:37:55</CreateDate>
                              <ModifyDate>2010-03-
17T19:37:53</ModifyDate>
                              <CreateUser>Bob</CreateUser>
                              <ModifyUser>Bob</ModifyUser>
                              <UserGroup>All</UserGroup>
                              <Permissions>
                               <Owner>write</Owner>
                               <Group>read</Group>
                               <All>read</All>
                              </Permissions>
                        </Security>
                        <Location d6p1:iid="4DET391B" d6p1:eid="-
224161716" xmlns="" />
                        <q5:NodeBID>-1</q5:NodeBID>
                        <q5:NodeBEquipmentType d6p1:iid="Node B with TMA"
d6p1:eid="667775795" />
                        <q5:PLMN d6p1:iid="PLMN0" d6p1:eid="690711692" />
                        <q5:Resources>
                              <q5:Resource>
                               <ULTotal xmlns="">10000</ULTotal>
                               <DLTotal xmlns="">10000</DLTotal>
                               <ULPriority xmlns="">5000</ULPriority>
                               <DLPriority xmlns="">5000</DLPriority>
                               <ULHandover xmlns="">5000</ULHandover>
                               <DLHandover xmlns="">5000</DLHandover>
                              </q5:Resource>
                        </q5:Resources>

      <q5:AllowedOperations>ReadWrite</q5:AllowedOperations>
                  </AppData>
                  <AppData
xmlns:q6="http://www.aircominternational.com/Schemas/UMTS/2009/05"
xsi:type="q6:NodeBType" d6p1:iid="NodeBC" d6p1:bvid="Northeast"
d6p1:eid="1017"
xmlns:d6p1="http://www.aircominternational.com/Schemas/CommonTypes/2009/0
5">
                        <Security xmlns="">
                              <CreateDate>2010-03-
17T19:37:55</CreateDate>
                              <ModifyDate>2010-03-
17T19:37:55</ModifyDate>
                              <CreateUser>Bob</CreateUser>
                              <ModifyUser>Bob</ModifyUser>
                              <UserGroup>All</UserGroup>
                              <Permissions>
                               <Owner>write</Owner>
                               <Group>read</Group>
                               <All>read</All>
                              </Permissions>
                        </Security>
                        <Location d6p1:iid="4DET391B" d6p1:eid="-
224161716" xmlns="" />
                        <q6:NodeBID>-1</q6:NodeBID>
                        <q6:NodeBEquipmentType d6p1:iid="Node B with TMA"
d6p1:eid="667775795" />
                        <q6:PLMN d6p1:iid="PLMN0" d6p1:eid="690711692" />
                        <q6:Resources>
                              <q6:Resource>
                               <ULTotal xmlns="">10000</ULTotal>
                               <DLTotal xmlns="">10000</DLTotal>
                               <ULPriority xmlns="">5000</ULPriority>
                               <DLPriority xmlns="">5000</DLPriority>
                               <ULHandover xmlns="">5000</ULHandover>
                               <DLHandover xmlns="">5000</DLHandover>
                              </q6:Resource>
                        </q6:Resources>

      <q6:AllowedOperations>ReadWrite</q6:AllowedOperations>
                  </AppData>
            </Data>
      </QueryResult>
</QueryResponse>
```

Note: The EDS has created default resources.

Neighbour Type Objects "Operation" Attribute

Here is an example of Soap envelope using the 'Operation' attribute:

```
<soapenv:Envelope >
   <soapenv:Body>
      <ns:Modify>
         <ns:data>
            <ns1:itemID>A1B970A6-E0A4-4782-973D-3835AE656F92</ns1:itemID>
            <ns2:masterID>B1B970A6-E0A4-4782-973D-
3835AE656F93</ns2:masterID>
            <ns3:TimeoutOverride>300</ns3:TimeoutOverride>
            <ns3:RunSequentially>false</ns3:RunSequentially>
            <ns:ModifyItems ns:objectType="NeighbourType"
ns:SuppressLogicaltoPhysicalAntennaCorrection="true"
ns:SparseListUpdate="true" ns:CreateIfNotFound="false"
ns:IgnoreQueryAndDoBulkUpdate="true">
               <ns1:itemID>A1B970A6-E0A4-4782-973D-
3835AE656F91</ns1:itemID>
               <ns:Select>
                  <ns5:OverrideCacheDefault UseQueryCache="false"/>
               </ns:Select>
               <ns:NewData>
                  <ns4:AppData xsi:type="ns12:NeighbourType"
Operation="Delete" Technology="LTE" ns14:iid="eNodeB2B" ns14:bvid="Bob"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                     <ns12:OutwardNbrs CarrierID="1" NbrCellID="CDMABS0A"
Technology="CDMA">
                        <ns12:NbrCarrierID>1</ns12:NbrCarrierID>
                     </ns12:OutwardNbrs>

<ns12:AllowedOperations>ReadUpdateDelete</ns12:AllowedOperations>
                  </ns4:AppData>
               </ns:NewData>
            </ns:ModifyItems>
           <ns:ModifyItems ns:objectType="NeighbourType"
ns:SuppressLogicaltoPhysicalAntennaCorrection="true"
ns:SparseListUpdate="true" ns:CreateIfNotFound="false"
ns:IgnoreQueryAndDoBulkUpdate="true">
               <ns1:itemID>A1B970A6-E0A4-4782-973D-
3835AE656F92</ns1:itemID>
               <ns:Select>
                  <ns5:OverrideCacheDefault UseQueryCache="false"/>
               </ns:Select>
               <ns:NewData>
                  <ns4:AppData xsi:type="ns12:NeighbourType"
Operation="Create" Technology="LTE" ns14:iid="eNodeB2B" ns14:bvid="Bob"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                     <ns12:OutwardNbrs CarrierID="1" NbrCellID="CDMABS0A"
Technology="CDMA">
                        <ns12:NbrCarrierID>1</ns12:NbrCarrierID>
                        <ns12:Status>Live</ns12:Status>
                     </ns12:OutwardNbrs>

<ns12:AllowedOperations>ReadUpdateDelete</ns12:AllowedOperations>
                  </ns4:AppData>
               </ns:NewData>
            </ns:ModifyItems>
           <ns:ModifyItems ns:objectType="NeighbourType"
ns:SuppressLogicaltoPhysicalAntennaCorrection="true"
ns:SparseListUpdate="true" ns:CreateIfNotFound="false"
ns:IgnoreQueryAndDoBulkUpdate="true">
```

```
               <ns1:itemID>A1B970A6-E0A4-4782-973D-
3835AE656F93</ns1:itemID>
               <ns:Select>
                  <ns5:OverrideCacheDefault UseQueryCache="false"/>
               </ns:Select>
               <ns:NewData>
                  <ns4:AppData xsi:type="ns12:NeighbourType"
Operation="Update" Technology="LTE" ns14:iid="eNodeB2B" ns14:bvid="Bob"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                     <ns12:OutwardNbrs CarrierID="1" NbrCellID="CDMABS0A"
Technology="CDMA">
                        <ns12:NbrCarrierID>1</ns12:NbrCarrierID>
                        <ns12:Status>Planned</ns12:Status>
                     </ns12:OutwardNbrs>

<ns12:AllowedOperations>ReadUpdateDelete</ns12:AllowedOperations>
                  </ns4:AppData>
               </ns:NewData>
            </ns:ModifyItems>
         </ns:data>
      </ns:Modify>
   </soapenv:Body>
```

10 Securing EDS with HTTPS

Introduction

This chapter outlines how to secure a given EDS instance with a pair of HTTPS certificates (client and server), signed by a trusted root authority. When EDS is running HTTPS, a client must issue a defined certificate public key to the server and the server must be aware of the client key's existence in its local certificate store. If the server rejects the client credentials, it will issue an HTTP 403 header in response to the client and terminate the connection.

For the purposes of this document, the examples will be based on self-signed certificates. However, in a production environment, it is highly recommended to use certificates issued by a recognized root authority, either provided via your organization's network administrators or via a trusted 3rd party provider.

Note: TEOCO will not be responsible for generating or issuing certificates to customers.

The examples focus on adding HTTPS support to the following ports and endpoints:

- 8732 – EDS HTTP SOAP
- 8730 – EDS WSDL
- 8632 – EDS (Basic) HTTP SOAP
- 8630 – EDS (Basic) WSDL
- 8682 – EDS RESTful

Terms and Abbreviations:

FQDN	- Fully Qualified Domain Name
WCF	- Windows Communication Foundation

Preparing the Server/Client

In order to follow this document, you will require administrative access to both a Windows server hosting EDS as well as a client machine, for the purposes of simplicity we will assume both machines are running windows. The steps outlined in this document have been successfully attempted on Windows 7, 10 and Server 2012.

To create self-signed certificates, you will need access to the following utilities:

- MakeCert.exe
- pvk2pfx.exe

These utilities are available as part of the Microsoft Platform SDK (or any Visual Studio Distribution).

Alternatively, Appendix 3 at the end of this chapter has an example Powershell 4.0 script that can create the necessary self-signed certificates and install them on your server.

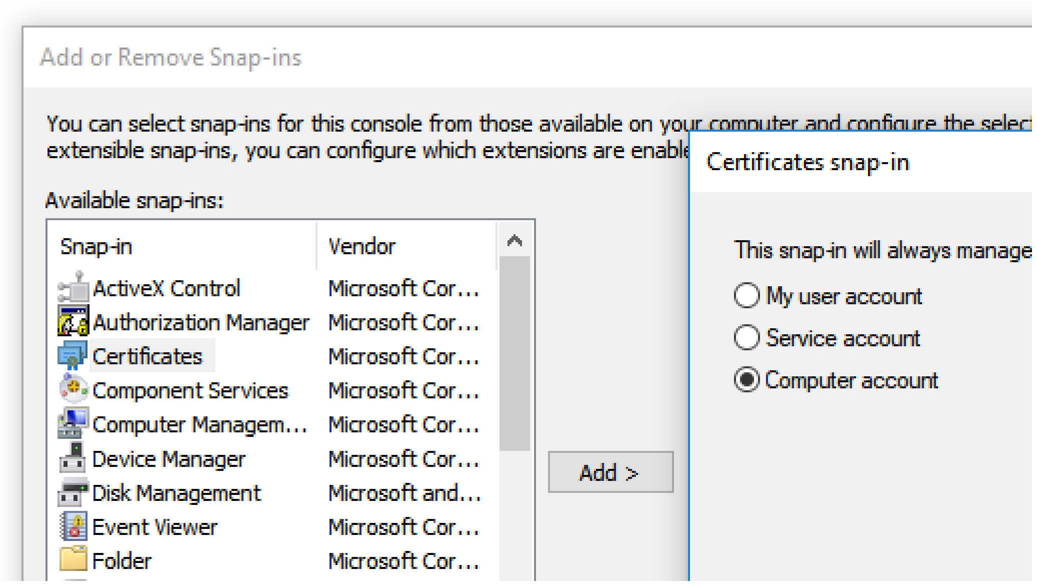
Note: It is beyond the scope of this document to provide guidance on using this script.

Opening the Local Machine Certificate Store

Opening the Local Machine Certificate Store

All certificates need to be imported into Windows Local Machine store via the Certificates snap-in for MMC (Microsoft Management Console). To open:

1. Run "mmc" from a console
2. Select "File->Add/Remove Snap-in…".
3. From the presented dialog select "Certificates".
4. Click "Add" and when asked which store you wish to open, select "Computer Account".


Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the select
extensible snap-ins, you can configure which extensions are enable

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microsoft Cor…
Authorization Manager	Microsoft Cor…
Certificates	Microsoft Cor…
Component Services	Microsoft Cor…
Computer Managem…	Microsoft Cor…
Device Manager	Microsoft Cor…
Disk Management	Microsoft and…
Event Viewer	Microsoft Cor…
Folder	Microsoft Cor…

Certificates snap-in

This snap-in will always manage

My user account [ ]
Service account [ ]
[✓ Computer account]

Creating a Trusted Root Certificate Authority

Before Windows will trust any clients linked to a self-signed certificate you need to create a chain of trust that the server hosting EDS will accept. This is achieved by establishing a root “Certificate Authority” and placing it within a trusted secure location on the server.

From an elevated command shell (launch cmd.exe as Administrator) run the steps below to create a root certificate. Then, using mmc import into the Local Computer “Trusted Root Certification Authorities\Certificates”, enter a password where prompted for the private key store.

Note: Having access to a private key allows consumers to create their own trusted keys, dependent on the private keys chain. Therefore you should always save your private keys in a secure location. Furthermore, by placing a self-signed (development) key within the Windows trusted CA store, this will force the server to trust any further certificates that are chained to it, potentially causing a security risk – so use this key with caution!

```
makecert.exe ^
-n "CN=DEMO Root CA - USE WITH CAUTION" ^
-r ^
-pe ^
-a sha512 ^
-len 4096 ^
-cy authority ^
```

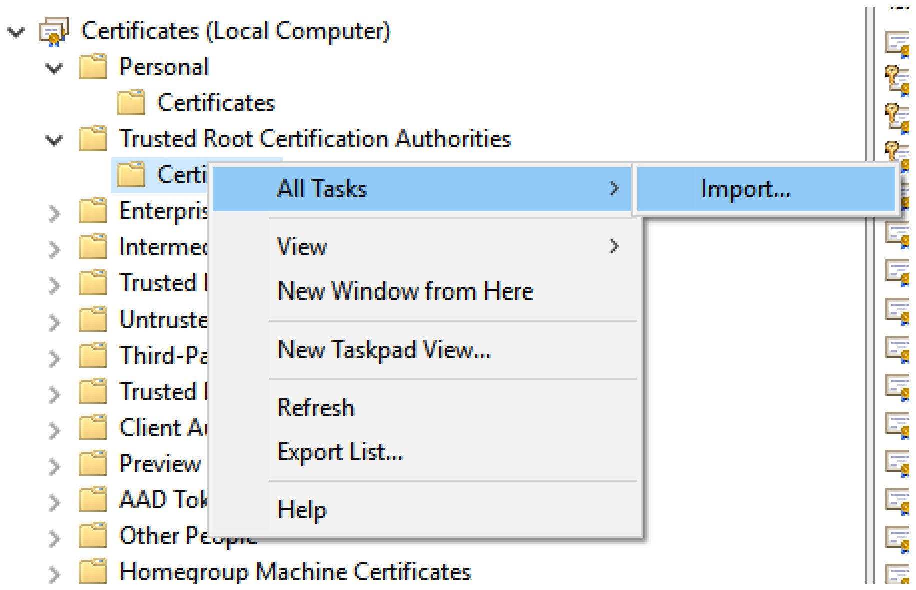
```
-sv DemoDevelopmentRootCA.Keystore.pvk ^
DemoDevelopmentRootCA.cer
```

Convert the private key store file (.pvk) into a portable .pfx file (personal exchange format). This file can be used by other SSL client technologies such as Java.

```
pvk2pfx.exe ^
-pvk DemoDevelopmentRootCA.Keystore.pvk ^
-spc DemoDevelopmentRootCA.cer ^
-pfx DemoDevelopmentRootCA.Keystore.pfx ^
-po <password>
```

To import the key into the trusted root store in mmc:

1. From the mmc window select “Trusted Root Certification Authorities->Certificates”.
2. Right click on the folder and select “All Tasks->Import”.
3. Follow the wizard until it asks for a file.
4. Select the .pfx file (as previous), and enter the password where prompted.



EDS Service Certificates

Once a root certificate has been established, it is necessary to create a service certificate to register with EDS.

Background

This paragraph is from the MSDN website:

Service certificates have the primary task of authenticating the server to clients. One of the initial checks when a client authenticates a server is to compare the value of the **Subject** field to the Uniform Resource Identifier (URI) used to contact the service: the DNS of both must match. For example, if the URI of the service is "`http://www.contoso.com/endpoint/`" then the **Subject** field must also contain the value "`www.contoso.com`".

Notes:

- The field can contain several values, each prefixed with an initialization to indicate the value. Most commonly, the initialization is "CN" for common name, for example, "CN = www.contoso.com". It is also possible for the Subject field to be blank, in which case the Subject Alternative Name field can contain the DNS Name value.
- The value of the Intended Purposes field of the certificate should include an appropriate value, such as "Server Authentication" or "Client Authentication".

Instructions

The steps required to create a service certificate are very similar to those for establishing the root CA, from a console execute the following:

```
makecert.exe ^
-n "CN=MyServer.demo.com" ^
-iv DemoDevelopmentRootCA.Keystore.pvk ^
-ic DemoDevelopmentRootCA.cer ^
-pe ^
-a sha512 ^
-len 4096 ^
-b 01/01/2017 ^
-e 01/01/2020 ^
-sky exchange ^
-eku 1.3.6.1.5.5.7.3.1 ^
-sv EDSServer.pvk ^
EDSServer.cer
```

Note: You must name the EDS service certificate after the fully qualified domain name of the host (FQDN). This would normally be <servername>.<dns path>

To verify you have the correct FQDN, you can ping your host address.

```
pvk2pfx.exe ^
-pvk EDSServer.pvk ^
-spc EDSServer.cer ^
-pfx EDSServer.pfx ^
-po password
```

Once the service certificate has been created, you must import the EDSServer.pfx certificate into the “Personal->Certificates” store using mmc following the same wizard steps as previously.

Client Certificates

A client certificate must also be created and imported into the EDS server, this is used to establish a mutual trust relationship with any approved EDS client that is permitted to connect. The same client certificate may then be shared with more than one client machine. Within the context of this example, any client machine will also need its own copy of the root certificate installed in order to support the same chain of trust.

To create a client certificate:

makecert.exe ^
-n "CN=demo.com" ^
-iv DemoDevelopmentRootCA.Keystore.pvk ^
-ic DemoDevelopmentRootCA.cer ^
-pe ^

```
-a sha512 ^
-len 4096 ^
-b 01/01/2017 ^
-e 01/01/2020 ^
-sky exchange ^
-eku 1.3.6.1.5.5.7.3.2 ^
-sv EDSClient.pvk ^
EDSClient.cer
```

Note: Unlike the service certificate, the client certificate can just use the domain name (without the server identity) in order for it to be shared with other machines within the local network. Replace the “demo.com” with your local network DNS path.

```
pvk2pfx.exe ^
-pvk EDSClient.pvk ^
-spc EDSClient.cer ^
-pfx EDSClient.pfx ^
-po password
```

Once the client certificate has been created, you must import the EDSClient.pfx certificate into the “Personal->Certificates” store using mmc following the same wizard steps as previous.

Registering the HTTPS Ports with Windows

Once the client and server certificates have been created and installed, you need to configure Windows to accept one of them on a set of preconfigured ports. To do this you will need to obtain the "thumbprint" of the service certificate.

Obtaining a Service Thumbprint

To obtain the thumbprint using mmc, find the EDS Service certificate, double click upon it and select the “Details” tab. From this tab scroll down to the “Thumbprint” item and copy the contents to the clipboard and paste into a text editor. This dialog supports a Unicode character set so when copying the thumbprint contents to a text editor or the command shell you may have hidden characters (at the front) that interfere with it. To remove them use an editor that allows you to convert the string to an ANSI charset. Once the thumbprint has been copied to a text editor, remove any spaces between the hexadecimal numbers.

Instructions

Using the thumbprint previously copied, run the following from the command shell:

```
netsh http add sslcert ipport=0.0.0.0:8732
certhash=bfa6d64854d530016c96c2e326926919cbc40c38 appid={08F6C9E4-DA87-
4C3D-86C4-CA0E272D55A4}

netsh http add sslcert ipport=0.0.0.0:8730
certhash=bfa6d64854d530016c96c2e326926919cbc40c38 appid={08F6C9E4-DA87-
4C3D-86C4-CA0E272D55A5}

netsh http add sslcert ipport=0.0.0.0:8630
certhash=bfa6d64854d530016c96c2e326926919cbc40c38 appid={08F6C9E4-DA87-
4C3D-86C4-CA0E272D55A6}

netsh http add sslcert ipport=0.0.0.0:8632
certhash=bfa6d64854d530016c96c2e326926919cbc40c38 appid={08F6C9E4-DA87-
4C3D-86C4-CA0E272D55A7}
```

```
netsh http add sslcert ipport=0.0.0.0:8682
certhash=bfa6d64854d530016c96c2e326926919cbc40c38 appid={08F6C9E4-DA87-
4C3D-86C4-CA0E272D55A8}
```

Where certhash is the thumbprint value from the Service Certificate (without spaces) and appid is taken from a Registry format generated GUID. The four appid GUIDs should be generated using the UUIDGEN tool included in the Windows SDK. Alternatively use the Windows PowerShell command "[guid]::NewGuid()" to generate four new GUIDs, remembering to add the braces at the start and end of the GUIDs.

After the thumbprints have been allocated, you will next need to reserve the https namespaces for same ports and their access rights. To do so, execute the following from the command line:

```
netsh http add urlacl url=https://+:8732/ user=Everyone
netsh http add urlacl url=https://+:8730/ user=Everyone
netsh http add urlacl url=https://+:8632/ user=Everyone
netsh http add urlacl url=https://+:8630/ user=Everyone
netsh http add urlacl url=https://+:8682/ user=Everyone
```

Configuring EDS to Use HTTPS

This section focuses on the steps required to configure EDS to use the newly created service certificate created earlier. The changes involve creating a new "binding" to replace the existing http binding as well as specifying where in the certificate store EDS can find our new certificate. Before starting, it is recommended to take a backup copy of the "TEOCO.EDS.Presentation.Runtime.exe.config" and/or "TEOCO.EDS.Composite.Runtime.exe.config" file.

Create an HTTPS Binding

Inside the config file, locate the "<system.serviceModel><bindings><basicHttpBinding>…" section. Beneath the existing binding for "EDS Streaming Binding" add the following entry:

```
<binding name="EDS Secure Streaming Binding" closeTimeout="00:03:00"
openTimeout="00:01:00" receiveTimeout="00:30:00" sendTimeout="00:30:00"
maxBufferSize="2147483647" maxBufferPoolSize="2147483647"
maxReceivedMessageSize="2147483647" messageEncoding="Mtom">
     <security mode="Transport">
          <transport clientCredentialType="Certificate"
           proxyCredentialType="None" realm=""/>
     </security>
<readerQuotas maxDepth="32" maxStringContentLength="2147483647"
maxArrayLength="2147483647" maxBytesPerRead="2147483647"
maxNameTableCharCount="2147483647"/>
</binding>
```

For the astute reader, this new binding is identical to the existing one except now it includes a nested "security" xml element. This new element is what instructs EDS to use transport level security (TLS) rather than the default method.

Create a Secure Service Behaviour

Service "behaviours" are a mechanism within the WCF framework that allow an endpoint additional customisation beyond their initial configuration. Within the same config file, locate the section named "<system.serviceModel><behaviors><serviceBehaviors>…".

Within this section add the following (after the existing behaviours):

```
<behavior name="TEOCO.EDS.v1.Secure">
    <serviceDebug includeExceptionDetailInFaults="true" />
    <serviceCredentials>
    <clientCertificate>
            <authentication certificateValidationMode="PeerTrust"
             trustedStoreLocation="LocalMachine"/>
    </clientCertificate>
    <serviceCertificate findValue="
bfa6d64854d530016c96c2e326926919cbc40c38"
     storeLocation="LocalMachine" storeName="My" x509FindType=
"FindByThumbprint"/>
</serviceCredentials>
</behavior>
```

Within this new entry you will need to indicate the EDS server name and its thumbprint to match the same value from the EDS Server Certificate – replace the string with the same value used in the previous section.

Update the EDS Endpoint Configurations with the New Behaviour and Binding

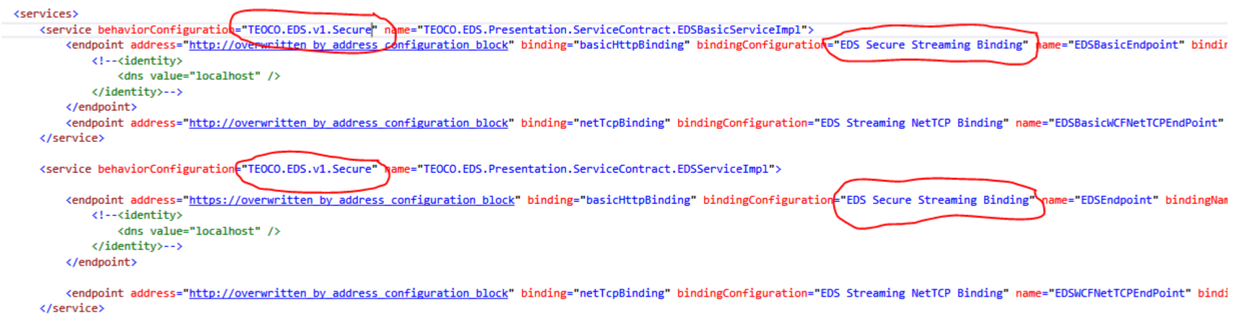
The next step is to switch over the two http EDS endpoints to use the new configuration:

Within the config file, locate the section named "services" and where highlighted substitute:

`"TEOCO.EDS.v1" for "TEOCO.EDS.v1.Secure"`

`- and -`

`"EDS Streaming Binding" for "EDS Secure Streaming Binding":`

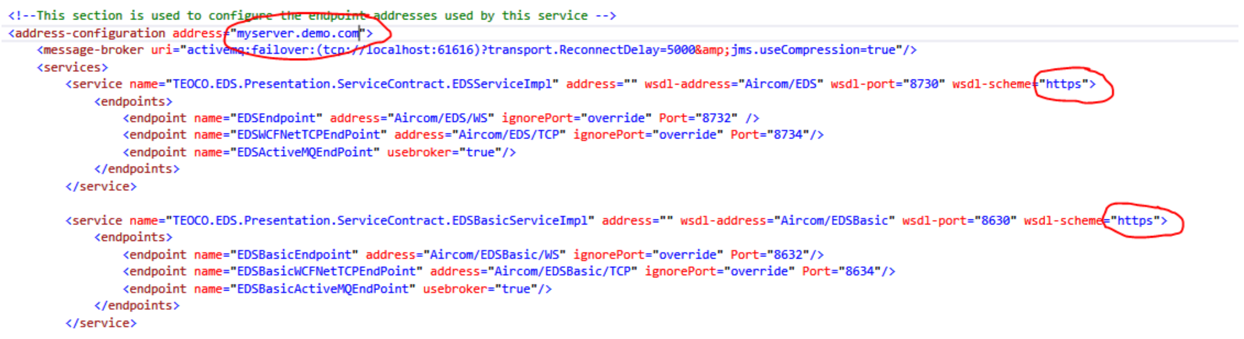


```
<services>
    <service behaviorConfiguration="TEOCO.EDS.v1.Secure" name="TEOCO.EDS.Presentation.ServiceContract.EDSBasicServiceImpl">
        <endpoint address="http://overwritten by address configuration block" binding="basicHttpBinding" bindingConfiguration="EDS Secure Streaming Binding" name="EDSBasicEndpoint" bindir
            <!--<identity>
                <dns value="localhost" />
            </identity>-->
        </endpoint>
        <endpoint address="http://overwritten by address configuration block" binding="netTcpBinding" bindingConfiguration="EDS Streaming NetTCP Binding" name="EDSBasicWCFNetTCPEndPoint"
    </service>

    <service behaviorConfiguration="TEOCO.EDS.v1.Secure" name="TEOCO.EDS.Presentation.ServiceContract.EDSServiceImpl">

        <endpoint address="https://overwritten by address configuration block" binding="basicHttpBinding" bindingConfiguration="EDS Secure Streaming Binding" name="EDSEndpoint" bindingNam
            <!--<identity>
                <dns value="localhost" />
            </identity>-->
        </endpoint>

        <endpoint address="http://overwritten by address configuration block" binding="netTcpBinding" bindingConfiguration="EDS Streaming NetTCP Binding" name="EDSWCFNetTCPEndPoint" bindi
    </service>
```

Note: it is also advisable to comment out the "identity" element within each block.

Update the WSDL Scheme

Finally, the last step is to alter the WSDL scheme near the top of the configuration file. Where indicated, replace "http" with "https" – also remember to set the correct hostname within this configuration section:



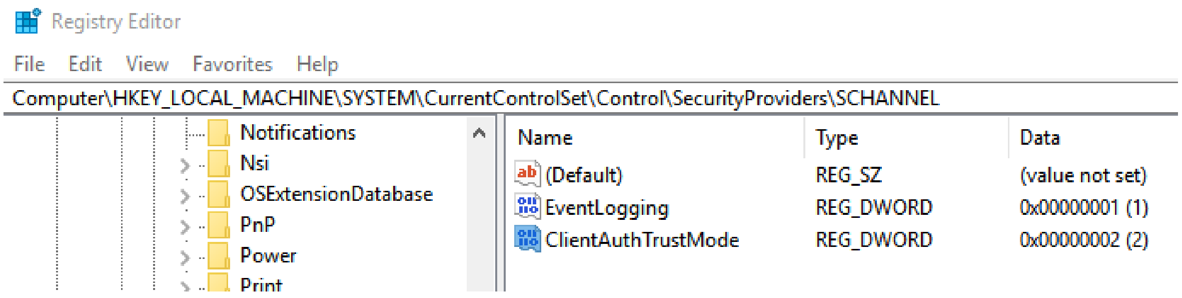
```
<!--This section is used to configure the endpoint addresses used by this service -->
<address-configuration address="myserver.demo.com">
    <message-broker uri="activemq:failover:(tcp://localhost:61616)?transport.ReconnectDelay=5000&amp;jms.useCompression=true"/>
    <services>
        <service name="TEOCO.EDS.Presentation.ServiceContract.EDSServiceImpl" address="" wsdl-address="Aircom/EDS" wsdl-port="8730" wsdl-scheme="https">
            <endpoints>
                <endpoint name="EDSEndpoint" address="Aircom/EDS/WS" ignorePort="override" Port="8732" />
                <endpoint name="EDSWCFNetTCPEndPoint" address="Aircom/EDS/TCP" ignorePort="override" Port="8734"/>
                <endpoint name="EDSActiveMQEndPoint" usebroker="true"/>
            </endpoints>
        </service>

        <service name="TEOCO.EDS.Presentation.ServiceContract.EDSBasicServiceImpl" address="" wsdl-address="Aircom/EDSBasic" wsdl-port="8630" wsdl-scheme="https">
            <endpoints>
                <endpoint name="EDSBasicEndpoint" address="Aircom/EDSBasic/WS" ignorePort="override" Port="8632"/>
                <endpoint name="EDSBasicWCFNetTCPEndPoint" address="Aircom/EDSBasic/TCP" ignorePort="override" Port="8634"/>
                <endpoint name="EDSBasicActiveMQEndPoint" usebroker="true"/>
            </endpoints>
        </service>
```

Configure the SCHANNEL Registry Key to Permit Clients Registered in your Certificate Store

Conventionally, this step may not be required on older operating systems (such as Windows 7, 2008), but for more modern Operating Systems (10, 2012, 2016) it is necessary to alter where Windows looks for trusted client certificates.

Using Registry Editor, add a new DWORD key named "ClientAuthTrustMode" to the following path, set to "2":

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCH
ANNEL
```



Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL

Name	Type	Data
(Default)	REG_SZ	(value not set)
EventLogging	REG_DWORD	0x00000001 (1)
ClientAuthTrustMode	REG_DWORD	0x00000002 (2)

Note: This step was necessary in the case of a self-signing demo, however be aware that it may not be required with a properly issued Root CA or client or server certificate. Without applying this change, any client that accesses the secured EDS endpoint will be issued with a simple "HTTP 403" rejection which can indicate anything from an unpermitted client credential or a misconfiguration of the server.

For a full explanation of this registry setting please refer to the following Microsoft article – "Schannel SSP Technical Overview":

https://technet.microsoft.com/en-us/library/dn786429(v=ws.11).aspx

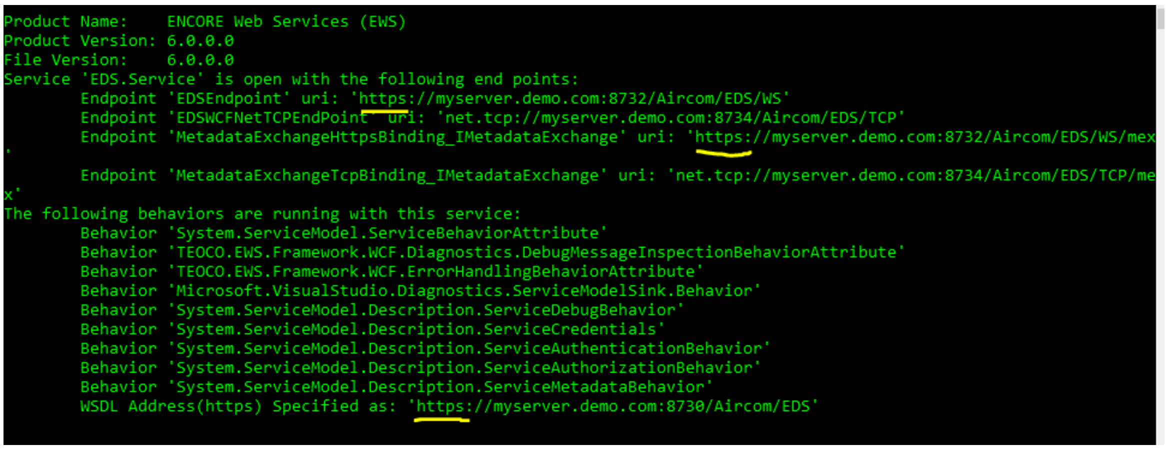
Testing the SOAP Interface

This section outlines some simple validation steps that can be done to confirm you have successfully configured EDS to support HTTPS. It will go as far as demonstrating how to use SOAP UI as a client to an EDS HTTPS instance however any advice on client-side code required to share an HTTPS certificate is beyond the scope of this document. TEOCO will at some future release an update to the EDS samples to include a C# based example client.

Confirm EDS Starts (Interactively)

Once all the configuration changes have been made to the TEOCO.EDS.Presentation.Runtime.config it is highly advisable to start the presentation service manually. This activity will quickly reveal any configuration issues that may have occurred. To start the presentation service manually, from a command shell (with administrator rights) enter: "TEOCO.EDS.Presentation.Runtime.exe".

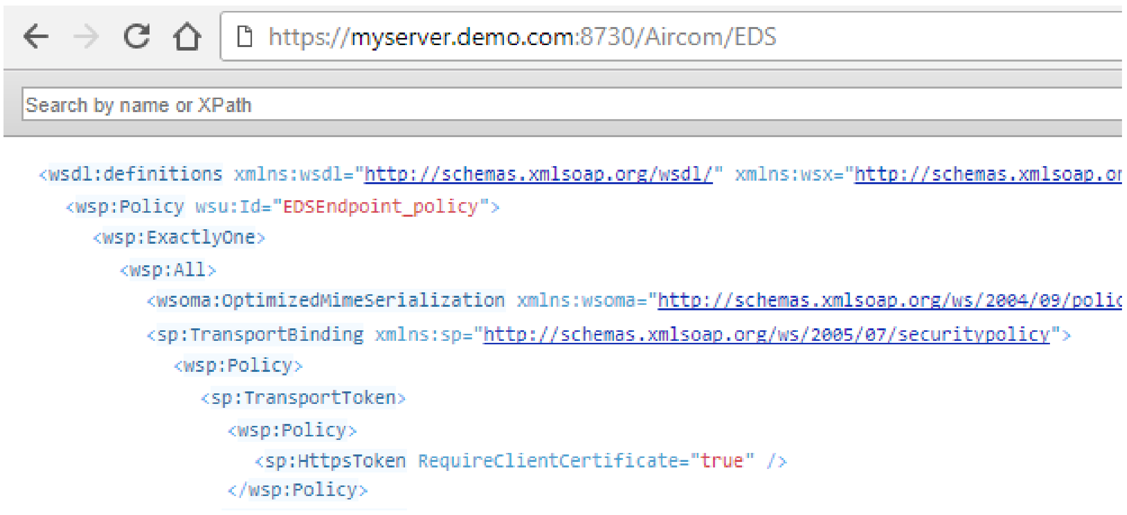
If EDS has started successfully you will see that the binding configuration listed now supports https:



```
Product Name:    ENCORE Web Services (EWS)
Product Version: 6.0.0.0
File Version:    6.0.0.0
Service 'EDS.Service' is open with the following end points:
        Endpoint 'EDSEndpoint' uri: 'https://myserver.demo.com:8732/Aircom/EDS/WS'
        Endpoint 'EDSWCFNetTCPEndPoint' uri: 'net.tcp://myserver.demo.com:8734/Aircom/EDS/TCP'
        Endpoint 'MetadataExchangeHttpsBinding_IMetadataExchange' uri: 'https://myserver.demo.com:8732/Aircom/EDS/WS/mex
'
        Endpoint 'MetadataExchangeTcpBinding_IMetadataExchange' uri: 'net.tcp://myserver.demo.com:8734/Aircom/EDS/TCP/me
x'
The following behaviors are running with this service:
        Behavior 'System.ServiceModel.ServiceBehaviorAttribute'
        Behavior 'TEOCO.EWS.Framework.WCF.Diagnostics.DebugMessageInspectionBehaviorAttribute'
        Behavior 'TEOCO.EWS.Framework.WCF.ErrorHandlingBehaviorAttribute'
        Behavior 'Microsoft.VisualStudio.Diagnostics.ServiceModelSink.Behavior'
        Behavior 'System.ServiceModel.Description.ServiceDebugBehavior'
        Behavior 'System.ServiceModel.Description.ServiceCredentials'
        Behavior 'System.ServiceModel.Description.ServiceAuthenticationBehavior'
        Behavior 'System.ServiceModel.Description.ServiceAuthorizationBehavior'
        Behavior 'System.ServiceModel.Description.ServiceMetadataBehavior'
        WSDL Address(https) Specified as: 'https://myserver.demo.com:8730/Aircom/EDS'
```

Confirm that the HTTPS WSDL Endpoint Responds

Once EDS has started, you should now be able to access the secured HTTPS WSDL endpoint. Using a web browser such as Google chrome, enter the URL as indicated by your configured WSDL address:



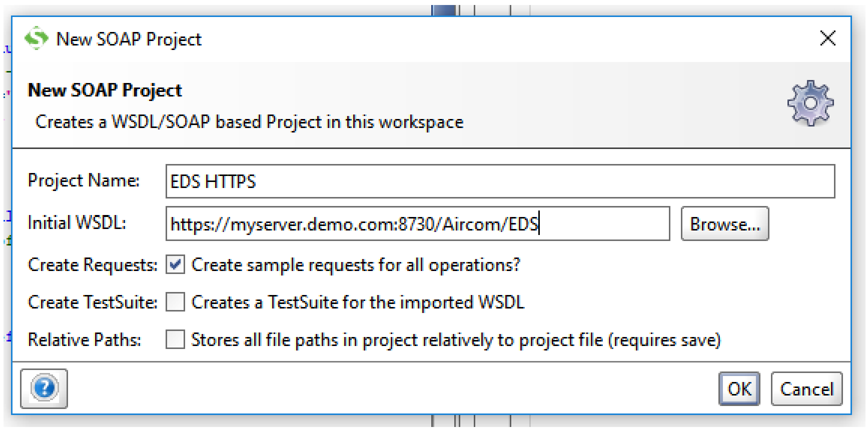
```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsx="http://schemas.xmlsoap.or
  <wsp:Policy wsu:Id="EDSEndpoint_policy">
    <wsp:ExactlyOne>
      <wsp:All>
        <wsoma:OptimizedMimeSerialization xmlns:wsoma="http://schemas.xmlsoap.org/ws/2004/09/polic
        <sp:TransportBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
          <wsp:Policy>
            <sp:TransportToken>
              <wsp:Policy>
                <sp:HttpsToken RequireClientCertificate="true" />
              </wsp:Policy>
```

You should be able to see your secured endpoint WSDL in response. You must use the exact configured hostname as per your SSL certificate otherwise your browser may reject the site as a security risk.

Test a SOAP Call Using SoapUI

SoapUI is an excellent third party opensource application for testing and debugging web services, it is available from the following link: https://www.soapui.org/

Once installed you can quickly create a new project by clicking on "File->New SOAP Project" and entering the EDS HTTP WSDL address it will download the API signatures and create test method calls for each of the available operations:



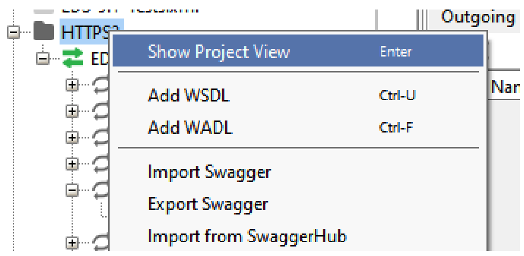
New SOAP Project

New SOAP Project
Creates a WSDL/SOAP based Project in this workspace

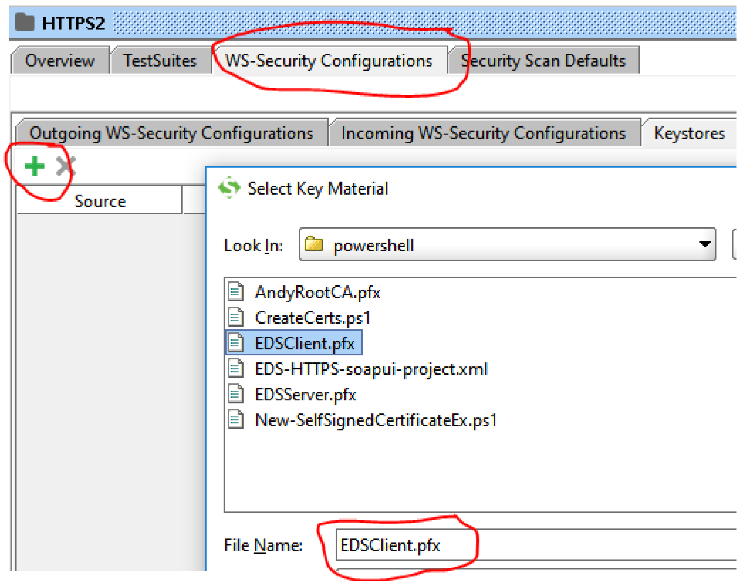
Project Name: EDS HTTPS
Initial WSDL: https://myserver.demo.com:8730/Aircom/EDS
Create Requests: [✓] Create sample requests for all operations?
Create TestSuite: [ ] Creates a TestSuite for the imported WSDL
Relative Paths: [ ] Stores all file paths in project relatively to project file (requires save)

As we are using a secured binding we need to give SoapUI our client certificate that we created earlier. This certificate is used to provide a client credential to EDS in order for it to trust the client.

From the project view on the left-hand side, right click on required project->Show Project View.



In WS-Security Configurations\Keystores tab click on +.



HTTPS2
Overview
TestSuites
WS-Security Configurations
Security Scan Defaults
Outgoing WS-Security Configurations
Incoming WS-Security Configurations
Keystores
Source
Select Key Material

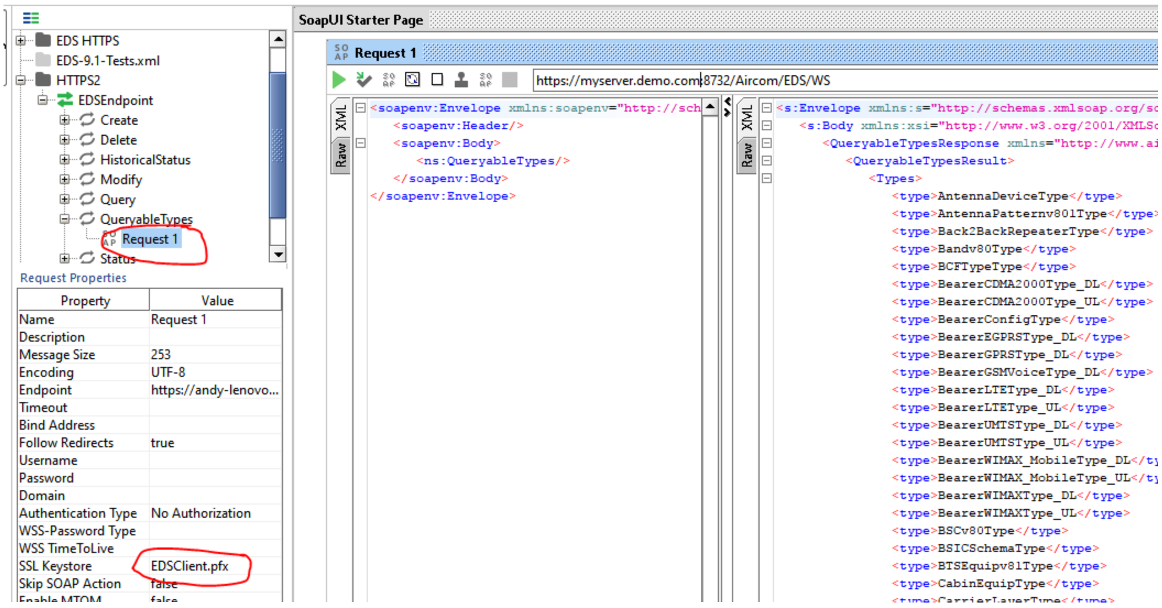
Look In: powershell

AndyRootCA.pfx
CreateCerts.ps1
EDSClient.pfx
EDS-HTTPS-soapui-project.xml
EDSServer.pfx
New-SelfSignedCertificateEx.ps1

File Name: EDSClient.pfx

Navigate to the EDSClient.PFX file that was created earlier and and supply password for certificate.

Then on the individual Request (QueryableTypes is recommended), select "SSL Keystore" in the Properties window and select the client certificate "EDSClient.pfx". Finally select the green "play" button on the SOAP operation and the API should respond similar to the following:



```
EDS HTTPS
EDS-9.1-Tests.xml
HTTPS2
  EDSEndpoint
    Create
    Delete
    HistoricalStatus
    Modify
    Query
    QueryableTypes
      Request 1
    Status
```

Request Properties

Property	Value
Name	Request 1
Description	
Message Size	253
Encoding	UTF-8
Endpoint	https://andy-lenovo...
Timeout	
Bind Address	
Follow Redirects	true
Username	
Password	
Domain	
Authentication Type	No Authorization
WSS-Password Type	
WSS TimeToLive	
SSL Keystore	EDSClient.pfx
Skip SOAP Action	false
Enable MTOM	false

https://myserver.demo.com:8732/Aircom/EDS/WS

```
<soapenv:Envelope xmlns:soapenv="http://sch
   <soapenv:Header/>
   <soapenv:Body>
      <ns:QueryableTypes/>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/so
   <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSc
      <QueryableTypesResponse xmlns="http://www.ai
         <QueryableTypesResult>
            <Types>
               <type>AntennaDeviceType</type>
               <type>AntennaPatternv801Type</type>
               <type>Back2BackRepeaterType</type>
               <type>Bandv80Type</type>
               <type>BCFTypeType</type>
               <type>BearerCDMA2000Type_DL</type>
               <type>BearerCDMA2000Type_UL</type>
               <type>BearerConfigType</type>
               <type>BearerEGPRSType_DL</type>
               <type>BearerGPRSType_DL</type>
               <type>BearerGSMVoiceType_DL</type>
               <type>BearerLTEType_DL</type>
               <type>BearerLTEType_UL</type>
               <type>BearerUMTSType_DL</type>
               <type>BearerUMTSType_UL</type>
               <type>BearerWIMAX_MobileType_DL</ty
               <type>BearerWIMAX_MobileType_UL</ty
               <type>BearerWIMAXType_DL</type>
               <type>BearerWIMAXType_UL</type>
               <type>BSCv80Type</type>
               <type>BSICSchemaType</type>
               <type>BTSEquipv81Type</type>
               <type>CabinEquipType</type>
               <type>CarrierLayerType</type>
```

```
. .\CreateCerts.ps1 -hostname myserver -dns demo.com
```

Note: SoapUI sometimes caches previous responses from a web service so its advisable to select the correct certificate and restart before attempting an EDS query – otherwise it may return an HTTP 403.

Example TEOCO.Presentation.Runtime.Config

The following contains a fragment from an EDS config containing all of the changes listed from within this document. This example also contains changes to the netTCP binding which is used by the EDS Loader (by default):

```
<address-configuration address="MyServer.Demo.Com">
<message-broker uri="activemq: failover: (tep://localhost: 61616)
?transport
 Reconnect Delay=5000&amp;jms.useCompression =true" />
 <services>
      <service name="TEOCO. EDS. Presentation ServiceContract.EDSService
Impl"
 address="" wsdladdress="Aircom/EDS" wsdl-port="8730" wsdl-
scheme="https">
           <endpoints>
                <endpoint name="EDSEndpoint" address="Aircom/EDS/WS"
ignore
 Port="override" Port="8732" />
                <endpoint name="EDSWCFNet TCPEndPoint"
address="Aircom/EDS/TCP"
 ignore Port="override" port="8734" />
           </endpoints>
      </service>

      <service name="TEOCO. EDS. Presentation
ServiceContract.EDSBasicService Impl"
 address="" wsdladdress="Aircom/EDSBasic" wsdl-port="8630" wsdl-
```

```
scheme="https">
           <endpoints>
                <endpoint name="EDSBasicEndpoint"
address="Aircom/EDSBasic/WS"
 ignore Port="override" port="8632" />
                <endpoint name="EDSBasic CFNet TCPEndPoint"
 address="Aircom/EDSBasic/TCP" ignore Port="override" port="8634" />
           </endpoints>
      </service>

      <service name="TEOCO. EDS. Presentation ServiceContract.EDSRest
Service"
 port="8733" address="/Aircom/EDS/REST" disablewsdl="true" />
      <service name="TEOCO. EDS. Presentation Service
Contract.EDSBasicRest
 Service" port="8633" address="/Aircom/EDSBasic/REST" disablewsdl="true"
/>
      </services>
 </address-configuration>

 <system.serviceModel>

  <extensions>
       <behaviorExtensions
            <add name="jsonHttpBehaviour" type="TEOCO. EDS.
Presentation.XML.ASSET
.Helpers.JsonBehaviorElement, TEOCO. EDS. Presentation XML.ASSET" />
       </behaviorExtensions>
  </extensions>

 <bindings>
      <basicHttpBinding>
           <binding name="EDS Streaming Binding" closeTimeout="00:03:00"
 openTimeout="00:01:00" receiveTimeout="00:30:00" sendTimeout="00:30:00"
 maxBufferSize="2147483647" maxBufferPoolSize="2147483647"
 maxReceivedMessageSize="2147483647" messageEncoding="Mtom"
 transferMode="Streamed">
                <readerQuotas maxDepth="32"
maxStringContentLength="2147483647"
 maxArrayLength="2147483647" maxBytesPerRead="2147483647"
 maxNameTableCharCount="2147483647" />
           </binding>
           <binding name="EDS Secure Streaming Binding"
closeTimeout="00:03:00"
 openTimeout="00:01:00" receiveTimeout="00:30:00" sendTimeout="00:30:00"
 maxBufferSize="2147483647" maxBufferPoolSize="2147483647"
 maxReceivedMessageSize="2147483647" messageEncoding="Mtom">
                <security mode="Transport">
                      <transport clientCredentialType="Certificate"
 proxyCredentialType="None" realm=""/>
                </security
                <readerQuotas maxDepth="32" maxString
ContentLength="2147483647"
 maxArrayLength="2147483647" maxBytesPerRead="2147483647"
 maxNameTableCharCount="2147483647"/>
           </binding>
     </basicHttpBinding>
     <net TcpBinding>
           <binding name="EDS Streaming NetTCP Binding"
sendTimeout="00:30:00"
 transferMode="Streamed" maxReceivedMessageSize="2147483647">
                <readerQuotas maxStringContentLength="2147483647"
 maxArrayLength="2147483647" maxNameTableCharCount="2147483647" />
                <security mode="Transport">
                       <transport clientCredentialType="Certificate"/>
                </security>
            </binding> </netTcpBinding> <customBinding>
            <binding name="Custom HTTP Binding">
                <webMessageEncoding webContentTypeMapperType="TEOCO. EDS.
 Presentation XML.ASSET.Helpers. RawContentTypeMapper, TEOCO.EDS.
 Presentation XML ASSET" />
                <httpTransport maxReceivedMessageSize="2147483647"
 transferMode="Streamed" manualAddressing="true" />
            </binding>
        </customBinding>
 </bindings>

 <services>
      <service behaviorConfiguration="TEOCO. EDS.v1.Secure"
 name="TEOCO. EDS. Presentation Service Contract EDSBagicService Impl">
           <endpoint address="http://overwritten by address configuration
block"
 binding="basicHttpBinding" bindingConfiguration="EDS Secure Streaming
Binding"
 name="EDSBasicEndpoint"
bindingNamespace="http://www.aircominternational.com/
 contract/EDSBasic/2013/03" contract="TEOCO. EDS. Presentation.XML.ASSET.
 Contracts.IEDSBasicService"/>

           <endpoint address="http://overwritten by address configuration
block"
 binding="netTcpBinding" bindingConfiguration="EDS Streaming Net TCP
Binding"
 name="EDSBasicWCFNet TCPEndPoint"

bindingNamespace="http://www.aircominternational.com/contract/EDSBasic/20
13/03"
 contract="TEOCO. EDS. Presentation XML.ASSET. Contracts.
IEDSBasicService" />
      </service>

      <service behaviorConfiguration="TEOCO. EDS.vi. Secure"
 name="TEOCO. EDS. Presentation ServiceContract.EDSService Impl"
           <endpoint address="https://overwritten by address
configuration_block"
 binding="basicHttpBinding" bindingConfiguration="EDS Secure Streaming
Binding"
 name="EDSEndpoint"

bindingNamespace="http://www.aircominternational.com/contract/EDS/2009/05
"
 contract="TEOCO.EDS. Presentation XML.ASSET. Contracts. IEDSService" />

           <endpoint address="http://overwritten by address configuration
block"
 binding="netTcpBinding" bindingConfiguration="EDS Streaming Net TCP
Binding"
 name="EDSWCFNetTCPEndPoint"

bindingNamespace="http://www.aircominternational.com/contract/EDS/2009/05
"
 contract="TEOCO.EDS. Presentation.XML.ASSET. Contracts. IEDSService" />
      </service>

      <service behaviorConfiguration="TEOCO. EDS.V1"
 name="TEOCO.EDS. Presentation ServiceContract EDSRestService"
           <endpoint address="http://overwritten by address
configuration_block"
 binding="customBinding" bindingConfiguration="Custom HTTP Binding"
 behaviorConfiguration="RESTFriendly" name="EDSRESTEndpoint"
```

```
bindingNamespace="http://www.aircominternational.com/contract/EDS/2009/05
"
 contract="TEOCO.EDS. Presentation.XML.ASSET. Contracts.
IEDSRestService"/>
      </service>

      <service behaviorConfiguration="TEOCO. EDS.V1"
 name="TEOCO. EDS. Presentation ServiceContract EDSBasicRestService">
           <endpoint address="http://overwritten by address configuration
block"
 behaviorConfiguration="RESTFriendly" binding="customBinding"
 bindingConfiguration="Custom HTTP Binding" name="EDSBasicRESTEndpoint"

bindingNamespace="http://www.aircominternational.com/contract/EDS/2013/03
"
 contract="TEOCO.EDS. Presentation XML.ASSET. Contracts. IEDSBasicRest
Service"/>
      </service>
 </services>

 <behaviors>
      <endpoint Behaviors>
           <behavior name="RESTFriendly">
                <jsonHttpBehaviour />
           </behavior>
      </endpoint Behaviors>

      <serviceBehaviors
           <behavior name="TEOCO. EDS.V1">
                <serviceDebug includeExceptionDetailInFaults="true" />
           </behavior>
           <behavior name="TEOCO. EDS.V1.Secure">
                <serviceDebug includeExceptionDetailInFaults="true" />
                <serviceCredentials>
                    <clientCertificate>
                        <authentication
certificateValidationMode="PeerTrust"
                        trustedStoreLocation="LocalMachine"
                        customCertificateValidatorType=""/>
                    </clientCertificate>
                    <serviceCertificate
 findValue="4102438896512079l0FE6C712BF5A86AA1222E33" store
Location="LocalMachine"
 storeName="My" x509FindType="FindByThumbprint"/>
                </serviceCredentials>
           </behavior>
      </serviceBehaviors>
 </behaviors>
 <diagnostics>
      <messageLogging logEntireMessage="false" logMalformed
Messages="false"
 logMessagesAtTransport Level="false" />
 </diagnostics
</system.serviceModel>
```

Example WCF Client Binding

The following fragment describes what a WCF client configuration might contain in order to share a certificate with a secured EDS endpoint (either HTTPS or netTCP). The client must share a certificate of its own to the server in order to be trusted and that the client certificate is installed on the server itself.

```
<system.serviceModel>
      <bindings>
           <basicHttpBinding>
                <binding name="EDSEndpoint"
maxBufferPoolSize="2147483647"
 maxReceivedMessageSize="2147483647"messageEncoding="Mtom">
                     <security mode="Transport">
                          <transport clientCredentialType="Certificate"
/>
                     </security
                 </binding>
           </basicHttpBinding>
           <net TcpBinding>
                 <binding name="EDS Streaming NetTCP Binding"
 sendTimeout="00:30:00" transferMode="Streamed"
 maxReceivedMessageSize="2147483647">
                      <readerQuotas maxStringContentLength="2147483647"
 maxArrayLength="2147483647" maxNameTableCharCount="2147483647" />
                     <security mode="Transport">
                          <transport clientCredentialType="Certificate"/>
                     </security>
                 </binding>
           </netTcpBinding>
      </bindings>

      <behaviors>
           <endpoint Behaviors>
                <behavior name="TEOCO. EDS.V1.Secure">
                     <clientCredentials>
                          <clientCertificate
 findValue="4862B930291A7B57FA060917292198694751F0A5"
 store Location="LocalMachine" storeName="My" x509
 FindType="FindByThumbprint"/>

                          <serviceCertificate>
                              <authentication
 certificateValidationMode="PeerorChainTrust"
 trustedStoreLocation="LocalMachine"/>
                          </serviceCertificate>
                     </clientCredentials>
                     <callbackDebug includeExceptionDetailInFaults="true"
/>
                </behavior>
            </endpoint Behaviors>
      </behaviors>

      <client>
           <endpoint
address="https://myserver.demo.com:8732/Aircom/EDS/WS"
 binding="basicHttpBinding" bindingConfiguration="EDSEndpoint"
 contract="EDSHttps EDS" name="EDSEndpoint"
 behaviorConfiguration="TEOCO. EDS.vi.Secure" />
          <endpoint
address="net.tcp://myserver.demo.com:8734/Aircom/EDS/TCP"
 binding="netTcpBinding" bindingConfiguration="EDS Streaming Net TCP
```

```
Binding"
 contract="EDSHttps EDS" name="EDSWCENet TCPEndPoint"
 behaviorConfiguration="TEOCO.EDS.v1.Secure">
          </endpoint>
     </client>
</system.serviceModel>
```

Powershell 4.0 Script for Creating Self-signed Certificates

The following script is an alternative method for creating a self-signed CA and client server certificates for use with this guide. To use this script you must have a workstation with Powershell 4.0 installed.

Note: This script is provided for information purposes only and is not supported by TEOCO.

To execute the script from a Powershell command line, enter:

```
..\CreateCerts.psl -hostname myserver -dns demo.com
```

Where hostname is your EDS server and dns is the trailing FQDN suffix.

```
param (
     [string] $hostname = "my-server",
     [string] $dns = "demo.mydomain.com"
     [string] $organisation = "O=MYCOMPANY, C=UK"
)

$root = New-SelfSigned Certificate -HashAlgorithm sha384 -KeyAlgorithm
RSA
 -KeyLength 4096
-Subject "CNEEDS HTTPS Demo Root Authority, $organisation"
-KeyUsage Digital Signature, CertSign -NotAfter (get-date) AddYears (10)
-CertStoreLocation "Cert:\LocalMachine My" -Type Custom

$rootca = $root Thumbprint

$EDSServer = New-SelfSignedCertificate -Type Custom -Subject
 "CN=$hostname. $dns"
    -HashAlgorithm sha256 -KeyAlgorithm RSA -KeyLength 2048
    -KeyUsage KeyEncipherment, Digital Signature -KeyExport Policy
Exportable
    -Key Spec KeyExchange -Provider "Microsoft Enhanced RSA and AES
 Cryptographic Provider"
    -CertStore Location "cert:\LocalMachine My"
    -Signer cert:\LocalMachine My Srootca
    -TextExtension ("2.5.29.37={text}1.3.6.1.5.5.7.3.1","2.5.29.17={text}
 DNS=$hostname. $ang")

$eds = $EDSServer Thumbprint
$path = "cert:\localmachine\my\" + $EDSServer Thumbprint

Export-PfxCertificate -cert $path -FilePath - \EDSServer.pfx -Password
 (ConvertTo-SecureString "password" -As PlainText -Force)

$EDSClient = New-SelfSignedCertificate -Type Custom -Subject "CN=$dns"
    -HashAlgorithm sha256 -KeyAlgorithm RSA -KeyLength 2048
    -KeyUsage KeyEncipherment Digital Signature
    -CertStoreLocation "cert:\LocalMachine My"
    -Signer cert:\LocalMachine\My\$rootca
    -TextExtension ("2.5.29.37={text}1.3.6.1.5.5.7.3.2","2.5.29.17={text}
 DNS=$dns")
```

```
$path = "cert:\localmachine my\" + $EDSClient Thumbprint

Export-PfxCertificate -cert $path -FilePath . \EDSClient.pfx -Password
 (ConvertTo-SecureString "password" -AsPlainText -Force)

& netsh http delete sslcert ipport=0.0.0.0:8732
& netsh http delete sslcert ipport=0.0.0.0:8730

& netsh http add sslcert ipport=0.0.0.0:8732 certhash=$eds
 appid='{08F6C9E4-DA87-4C3D-86C4CAOE272D55A4}' client
certnegotiation=enable
& netsh http add salcert ipport=0.0.0.0:8730 certhash=$eds
 appid='{08F6C9E4-DA87-4C3D-86C4CADE272D55A5}' #client
certnegotiation=enable
```

11 Supported EDS Query Attributes

This table lists the attributes which can be queried against, and their applicable data types:

Attribute	Description	Supported Types
iid	Unique string ID	All
eid	Unique Database key value	All
bvid	Project Name	All
niid	New Unique string ID (used for renaming elements)	MU-Nodes, All Cell Types (rename) Neighbours (reparent)
CreateDate	Specified date	All
ModifyDate	Modified date	All
CreateUser	User who created the object	All except GSM SubCell, Cell (GSM, UMTS, LTE, WiFi, NR)
ModifyUser	User who made the latest change to the object	All
UserGroup	Group that has read/write permissions access to the object	All except GSM SubCell, Cell (GSM, UMTS, LTE, WiFi, NR)
Name1	First Name	Property, WMSC, MSC, BSC, GSM Subcell, Repeaters, SGSN, RNC, MUNode (GSM, UMTS, LTE, WiFi, NR), LTE MME, LTE SAEGW, PMP Hub
Name2	Second Name	Property, WMSC, MSC, BSC, GSM Subcell, Repeaters, SGSN, RNC, MUNode (GSM, UMTS, LTE, WiFi, NR), LTE MME, LTE SAEGW, PMP Hub
Comments	Comments	Property, WMSC, MSC, BSC, GSM Subcell, Repeaters, SGSN, RNC, MUNode (GSM, UMTS, LTE, WiFi, NR), LTE MME, LTE SAEGW, PMP Hub
Location	Property ID associated with a network element	WMSC, MSC, BSC, GSM Subcell, Repeaters, SGSN, RNC, PtPLinkEnd, MUNode (GSM, UMTS, LTE, WiFi, NR, LTE MME, LTE SAEGW, PMP Hub
Parent	Owner Parent ID	Cell (GSM, UMTS, LTE, WiFi, NR), Repeaters, GSM Subcell
CellName	Additional Cell ID	Cell (GSM, UMTS, LTE, WiFi, NR)
GSMEnabled	GSM technology enabled. (In earlier versions of EDS this attribute was "GSM" and supported GSM node).	MUNode, Cell
UMTSEnabled	UMTS technology enabled. (In earlier versions of EDS this attribute was "UMTS" and supported UMTS node).	MUNode, Cell
WIFIEnabled	Wi-Fi technology enabled. (In earlier versions of EDS this attribute was "WIFI" and supported WIFI node).	MUNode, Cell
LTEEnabled	LTE technology enabled. (In earlier versions of EDS this attribute was "LTE" and supported LTE node).	MUNode, Cell

Attribute	Description	Supported Types
NREnabled	5G NR technology enabled	MUNode, Cell
Multi-Tech Cell Parameters* * As v10 Cells embed the technology they support a subquery is required to isolate specific technology parameters.	Allows subquery of cellular parameters	Cell
GSMParams	GSM technology cell parameters	Cell
UMTSParams	UMTS technology cell parameters	Cell
WIFIParams	Wi-Fi technology cell parameters	Cell
LTEParams	LTE technology cell parameters	Cell
NRParams	5G NR technology cell parameters	Cell
Cells* * When the subquery field for a child cell is non-unique (such as ModifyDate), the attributes below (GSMCells, WIFICells, UMTSCells, LTECells) can be used to force the subquery to only consider that technology type.	Allows subquery of all child cells (For version 9.1 MUNode only)	MUNode, Property
GSMCells	Allows subquery of all child GSM cells (For version 9.1 MUNode only)	MUNode
WIFICells	Allows subquery of all child Wi-Fi cells (For version 9.1 MUNode only)	MUNode
UMTSCells	Allows subquery of all child UMTS cells (For version 9.1 MUNode only)	MUNode
LTECells	Allows subquery of all child LTE cells (For version 9.1 MUNode only)	MUNode
SubCells	Allows subquery of all child subcells	Cell (GSM)
GSMId	Numeric GSM ID	Cell (GSM)
MNC	Mobile Network Code	Cell (GSM)
MCC	Mobile Country Code	Cell (GSM)
BCC	Broadcast Colour Code	Cell (GSM)
RAC	Radio Access Code	Cell (GSM, UMTS)
LAC	Location Area Code	Cell (GSM, UMTS)
NSEI	Network Service Entity Identifier	Cell (GSM, UMTS)
CellEquipment	Cell Equipment ID	Cell (GSM, UMTS)
CGI	CGI code	Cell (GSM, UMTS)
BTS	BTS ID attached to a cell site	MUNode (GSM), GSM Repeater
Cabin	Cabin ID attached to a cell site	MUNode (GSM), GSM Repeater
Tower	Tower ID attached to a cell site	MUNode (GSM), GSM Repeater
SubCellId	Subcell ID	GSM Subcell
SubCellName	Subcell Name	GSM Subcell
Traffic	Traffic	GSM Subcell
SignalOffset	Signal Offset	GSM Subcell
SignalThreshold	Signal Threshold	GSM Subcell

Attribute	Description	Supported Types
TrafficWeight	Traffic Weight	GSM Subcell
OverthrowLoadThreshold	Overthrow Load Threshold	GSM Subcell
ActivationThreshold	HR ActivationThreshold	GSM Subcell
TransmitterOutputPower	Transmitter PA Output	GSM Subcell
FixedPAOutput	Fixed PA Output (ACP constraint)	GSM Subcell
AveragePowerDecrease	8-PSK Average Power Decrease	GSM Subcell
TotalAllocatedTRX	Total TRX Allocated	GSM Subcell
HSN	HSN (Hopping Sequence Number)	GSM Subcell
MAIOOffset	MAIO Offset	GSM Subcell
MAIOStep	MAIO Step	GSM Subcell
MALID	MALID	GSM Subcell
UMTSCellID	UMTS Cell ID	Cell (UMTS)
LocalCellID	Local Cell ID	Cell (UMTS)
SAC	Service Area Code	Cell (UMTS)
NodeBEquipmentType	NodeB Equipment Type	MUNode (UMTS)
PLMN	Owner PLMN ID	MUNode, Repeater
Address1	First address line for a Property	Property
Address2	Second address line for a Property	Property
Town	Town	Property
Province	Province	Property
State	State	Property
PostCode	Post Code	Property
PropertyCode	Property Code	Property
Longitude	Longitude	Property
Latitude	Latitude	Property
GroundHgt	Ground height	Property
SearchRad	Search Radius	Property
Technology	Neighbour Relationship Technology Type	Neighbour
DLGain	Down Link Gain	Repeaters
ULGain	Up Link Gain	Repeaters
ULNoiseFigure	Up Link Noise Figure	Repeaters
BTSFixed	BTS Equipment Fixed	GSM Repeater
OutputPower	Repeater Output Power	GSM Repeater
LTECellID	LTE Cell ID	Cell (LTE)
SuMimoDlSupport	SU-MIMO Supported DL Modulation	Cell (LTE, NR)
SuMimoRxElems	SU-MIMO Rx Element Count	Cell (LTE, NR)
SuMimoTxElems	SU-MIMO Tx Element Count	Cell (LTE, NR)
SuMimoUlSupport	SU-MIMO Supported UL Modulation	Cell (LTE, NR)
MuMimoDlSupport	MU-MIMO Downlink Supported	Cell (LTE, NR)
MuMimoDlTerms	MU-MIMO Downlink Terminals	Cell (LTE, NR)
MuMimoUlSupport	MU-MIMO Uplink Supported	Cell (LTE, NR)

Attribute	Description	Supported Types
MuMimoUlTerms	MU-MIMO Uplink Terminals	Cell (LTE, NR)
NbrLimitInterCdma	Inter Neighbour Limit CDMA	Cell (LTE)
NbrLimitInterGsm	Inter Neighbour Limit GSM	Cell (LTE)
NbrLimitInterUmts	Inter Neighbour Limit UMTS	Cell (LTE)
NbrLimitIntraInter	Inter-Frequency Neighbour Limit	Cell (LTE)
NbrLimitIntraIntra	Intra-Frequency Neighbour Limit	Cell (LTE)
SignallingOverhead	Signalling Overhead	Cell (LTE, NR)
Tac	Type Allocation Code	Cell (LTE, NR)
CSArea		Cell (NR)
CSRange		Cell (NR)
IndoorClutterLossSchema		Cell (NR)
NCI		Cell (NR)
BeamformingSupportDL		Cell (NR)
BeamformingSupportUL		Cell (NR)
BeamformingArrayElems		Cell (NR)
BeamFormerType		Cell (NR)
BeamFormerMethodUL		Cell (NR)
CarrierParameters	Allows subquery of NR Cell Carrier Parameters	Cell (NR)
CodeSchema		Cell Carrier (NR)
PhysicalCellID		Cell Carrier (NR)
NoiseFigure		Cell Carrier (NR)
CAEnabledAtCell		Cell Carrier (NR)
CAEnabledAtgNode		Cell Carrier (NR)
InterNodeCAEnabled		Cell Carrier (NR)
CACellID		Cell Carrier (NR)
LoadPercentageDL		Cell Carrier (NR)
NoiseRiseUL		Cell Carrier (NR)
Scheduler		Cell Carrier (NR)
MaxScheduledUsers		Cell Carrier (NR)
HighSINREnabled		Cell Carrier (NR)
SINRThreshold		Cell Carrier (NR)
MUGNonRayleighAdj		Cell Carrier (NR)
LineOfSightEnabled		Cell Carrier (NR)
MaxTxPower		Cell Carrier (NR)
PSSOffset		Cell Carrier (NR)
PDCCHOffset		Cell Carrier (NR)
PDSCHOffset		Cell Carrier (NR)
CSIRSOffset		Cell Carrier (NR)
RefSignalSNRThresholdEnabled		Cell Carrier (NR)
RefSignalSNRThreshold		Cell Carrier (NR)

Attribute	Description	Supported Types
TimingAdvancedEnabled		Cell Carrier (NR)
TimingAdvancedMode		Cell Carrier (NR)
MaxTA		Cell Carrier (NR)
MaxRange		Cell Carrier (NR)
RSRPOffsetEnabled		Cell Carrier (NR)
RSRPOffset		Cell Carrier (NR)
AdaptiveSUMIMOThreshold Enabled		Cell Carrier (NR)
AdaptiveSUMIMOTrafficSINRThresholdUL		Cell Carrier (NR)
AdaptiveSUMIMOTrafficSINRThresholdDL		Cell Carrier (NR)
MUMIMOThresholdEnabled		Cell Carrier (NR)
MUMIMOTrafficSINRThresholdUL		Cell Carrier (NR)
MUMIMOTrafficSINRThresholdDL		Cell Carrier (NR)
MaxModulationDL		Cell Carrier (NR)
BeamFormingThresholdEnabled		Cell Carrier (NR)
BeamFormingTrafficSINRThresholdUL		Cell Carrier (NR)
BeamFormingTrafficSINRThresholdDL		Cell Carrier (NR)
NodeID	LTE Node ID	MUNode (LTE)
Supplier		Antenna Device
Height, Width, Depth	Physical dimensions	Antenna Device
Gain, Frequency, FrontToBackRatio, CrossPolarDiscrim, GainType, TiltType, Polarisation, ConfigID, ElecBeamAdj, AntennaPatternAngle, DownTilt, HorizontalBW, VerticalBW, ElementArrayIndicator	Parameters	Antenna Pattern
Patterns	Allows subquery of all child patterns.	Antenna Device
ConnectionType	The network element type attached to the connection.	Network Connection
EndA	Name of the element attached to EndA	Network Connection, Logical Network Connection, Cellular Network Connection
EndB	Name of the element attached to EndB	As above.
EndAType	Object type at EndA to ensure a unique match for the network connection.	Network Connection, Logical Network Connection, Cellular Network Connection
EndBType	Object type at EndB to ensure a unique match for the network connection.	As above.