GridOps Management Suite 3.10

Temporary Elements Integration

Functional Specification

Document Version: 1.0

Updated: June, 2024

Table of Contents

1. REFERENCES 8
2. ABSTRACT 9
3. REQUIREMENTS 10
4. OVERVIEW 11
4.1. Functional Design 11
4.1.1. Data Exchange 11
4.1.2. Integrity Update 12
4.1.3. Update of the List of Relevant Feeders 13
4.1.4. Source Adapter State Machine 14
4.2. Architectural Design 16
5. DETAILED DESIGN 17
5.1. Component Design 17
5.1.1. TEI Source Subscriber 17
5.1.2. TEI Adapter Manager 17
5.1.3. TEI Integrity Update Coordinator 17
5.1.4. TEI Source Internal Model 17
5.1.5. TEI Destination Internal Model 17
5.1.6. Destination MMS Subscriber 17
5.1.7. TEI NMS Reader 18
5.1.8. TEI TES Reader 18
5.1.9. TEI Source Sending Queue 18
5.1.10. TEI Sender Communication Component 18
6. INTERFACES 19
6.1. Add Cuts Operation 19
6.1.1. Overview 19
6.1.2. API Operation Specification 19
6.1.3. API Operation Usage 19
6.1.4. Data Format 20
6.1.4.1. Header Format 20
6.1.4.2. Logical Format 20
6.1.4.3. Physical Format 21
6.2. Delete Cuts Operation 22
6.2.1. Overview 22

6.2.2. API Operation Specification 22
6.2.3. API Operation Usage 23
6.2.4. Data Format 23
6.2.4.1. Header Format 23
6.2.4.2. Logical Format 23
6.2.4.3. Physical Format 24
6.3. Get All Cuts Operation 25
6.3.1. Overview 25
6.3.2. API Operation Specification 25
6.3.3. API Operation Usage 25
6.3.4. Data Format 25
6.3.4.1. Header Format 25
6.3.4.2. Logical Format 26
6.3.4.3. Physical Format 26
6.4. Add Jumpers Operation 27
6.4.1. Overview 27
6.4.2. API Operation Specification 27
6.4.3. API Operation Usage 27
6.4.4. Data Format 28
6.4.4.1. Header Format 28
6.4.4.2. Logical Format 28
6.4.4.3. Physical Format 29
6.5. Delete Jumpers Operation 31
6.5.1. Overview 31
6.5.2. API Operation Specification 31
6.5.3. API Operation Usage 31
6.5.4. Data Format 32
6.5.4.1. Header Format 32
6.5.4.2. Logical Format 32
6.5.4.3. Physical Format 32
6.6. Get All Jumpers Operation 33
6.6.1. Overview 33
6.6.2. API Operation Specification 33
6.6.3. API Operation Usage 34
6.6.4. Data Format 34
6.6.4.1. Header Format 34
6.6.4.2. Logical Format 34

6.6.4.3. Physical Format ... 35
6.7. Get All Feeders Operation ... 35
6.7.1. Overview ... 35
6.7.2. API Operation Specification ... 35
6.7.3. API Operation Usage ... 36
6.7.4. Data Format ... 36
6.7.4.1. Header Format ... 36
6.7.4.2. Logical Format ... 36
6.7.4.3. Physical Format ... 37
7. ADDITIONAL CONSIDERATIONS ... 38
7.1. Quality and Testing ... 38
7.1.1. Test GetFeederIds Interface ... 38
7.1.1.1. Preconditions ... 38
7.1.1.2. Execution ... 38
7.1.2. Test GetCuts Interface ... 38
7.1.2.1. Preconditions ... 39
7.1.2.2. Execution ... 39
7.1.3. Test GetJumpers Interface ... 39
7.1.3.1. Preconditions ... 39
7.1.3.2. Execution ... 39
7.1.4. Test AddCuts Interface ... 39
7.1.4.1. Preconditions ... 39
7.1.4.2. Execution ... 39
7.1.5. Test DeleteCuts Interface ... 40
7.1.5.1. Preconditions ... 40
7.1.5.2. Execution ... 40
7.1.6. Test AddJumpers Interface ... 41
7.1.6.1. Preconditions ... 41
7.1.6.2. Execution ... 41
7.1.7. Test DeleteJumpers Interface ... 41
7.1.7.1. Preconditions ... 41
7.1.7.2. Execution ... 41
7.1.8. Test Model Update on Destination ... 42
7.1.8.1. Preconditions ... 42
7.1.8.2. Execution ... 42
7.1.9. Test Source Adapter Startup Without Destination Side ... 42
7.1.9.1. Preconditions ... 42

7.1.9.2. Execution 43
7.1.10. Test Source Adapter Startup With Destination Side 43
7.1.10.1. Preconditions 43
7.1.10.2. Execution 43
7.1.11. Test Sending and Deleting Cuts 44
7.1.11.1. Preconditions 44
7.1.11.2. Execution 44
7.1.12. Test Sending and Deleting Jumpers 44
7.1.12.1. Preconditions 44
7.1.12.2. Execution 44
7.1.13. Reconnect the Source Side After the Failure of the Destination Side 45
7.1.13.1. Preconditions 45
7.1.13.2. Execution 45
7.1.14. Test Reaction of the Source Adapter to Model Update on the Destination Side 45
7.1.14.1. Preconditions 45
7.1.14.2. Execution 45
7.2. Deployment and Configuration 46
7.2.1. Deployment 46
7.2.2. Configuration 46
7.3. Upgradability 48
7.4. Extensibility 48
7.5. Security 48
8. DEFINITIONS AND ABBREVIATIONS 49

Table of Figures

Figure 4.1 – Data Exchange in the normal working mode 12
Figure 4.2 – Integrity Update 13
Figure 4.3 – Update the list of Feeder IDs 14
Figure 4.4 – TEI Architecture 16

Table of Tables

Table 4.1 – Source Adapter State Machine 14
Table 6.1 – Add cuts – API Operation specification 19
Table 6.2 – Add cuts – API operation usage 19
Table 6.3 – Add cuts – logical data format 20
Table 6.4 – Add cuts – logical data format 21
Table 6.5 – Add cuts - Data specification in the physical format 22
Table 6.6 – Delete Cuts – API Operation specification 22
Table 6.7 – Get All Cuts – API operation usage 23
Table 6.8 – Delete Cuts – logical data format 23
Table 6.9 – Delete cuts – logical data format 24
Table 6.10 – Delete Cuts – Data specification in the physical format 24
Table 6.11 – Get All Cuts – API Operation specification 25
Table 6.12 – Get All Cuts – API operation usage 25
Table 6.13 – Get All Cuts – logical data format 26
Table 6.14 – Get All Cuts – Data specification in the physical format 26
Table 6.15 – Add jumpers – API Operation specification 27
Table 6.16 – Add jumpers – API operation usage 27
Table 6.17 – Add jumpers – logical data format 28
Table 6.18 – Add jumpers point – logical data format 28
Table 6.19 – Add jumpers – logical data format 29
Table 6.20 – Add jumpers - Data specification in the physical format 31
Table 6.21 – Get All Jumpers – API operation usage 31
Table 6.22 – Delete Jumpers – logical data format 32
Table 6.23 – Delete jumpers – logical data format 32
Table 6.24 – Delete Jumpers – Data specification in the physical format 33
Table 6.25 – Get All Jumpers – API Operation specification 33
Table 6.26 – Get All Jumpers – API operation usage 34
Table 6.27 – Get All Jumpers – logical data format 35
Table 6.28 – Get All Feeders – API Operation specification 35
Table 6.29 – Get All Feeders – API operation usage 36
Table 6.30 – Get All Feeders – logical data format 36
Table 7.1 – TEI Source WebService GetFeederIds Configuration 46
Table 7.2 – TEISource WebService Configuration for Cuts 47
Table 7.3 – TEISource WebService Configuration for Jumpers 47
Table 7.4 – MPI Destination WebService Configuration 48

1. REFERENCES

#	Title	Description
1.	EcoStruxure GridOps Management Suite 3.10 Enterprise Integration Platform - Functional Specification	The document represents a set of common integration principles applied to all baseline integration adapters.

2. ABSTRACT

EcoStruxure GridOps Management Suite is a family of solutions designed to help electric utilities in the operations and management of their grid. It is offered as EcoStruxure ADMS, EcoStruxure Grid Operation, EcoStruxure DERMS or EcoStruxure Energy Transmission Operation solutions, which share the same technology platform.

NOTE: The functionality described in this document applies to all solutions.

NOTE: Most images presented in this document are related to the EcoStruxure ADMS solution and should be used as an example. The images for other solutions may differ slightly.

The purpose of this document is to describe the exchange of temporary elements between the source and the destination system.

This integration is needed for the offers such as Standalone DERMS or Grid Operation, where temporary elements come from ADMS system.

3. REQUIREMENTS

ReqId	Comment
TEI-001	The integration will encompass the temporary elements between Source and Destination system. Initially, only jumpers and cuts will be transferred from Source to Destination. When talking about temporary elements (TEs) henceforth, only jumpers and cuts will be considered.
TEI-002	Source system will have a subset of TEs that it will send to the Destination system. This subset will be defined by feeders that TEs are applied to. Only TEs applied to the feeders that exist on the Destination system will be transferred to it from the Source system. If an equipment that TE is applied to belongs to a feeder that does not exist on the Destination system, the TE will not be processed. Feeders are identified by customID.
TEI-003	The mapping of the TEs between Source and Destination systems will be done by customIDs of the TEs, meaning that the matching TEs in the two systems must have identical customIDs (case sensitive).
TEI-004	Communication between the Source adapter and Destination adapter will be secure, implemented as REST APIs.
TEI-005	Integrity update mechanism will be implemented between Source and Destination adapter.
TEI-006	Source adapter will only send the data to the Destination side if both adapters are on Hot and Active sites. Source adapter will only attempt to send the data if both Source and Destination side are available. In case the Destination adapter is not available, Source adapter will not attempt to send the data.
TEI-007	Destination side will perform validations, in the sense that it will not apply the changes that cannot be applied (e.g. TE with customID that already exists, or TE that is applied to the elements that do not exist on the Destination side).
TEI-008	TEs placed on Destination system through this integration will all have the same user, which will represent the Source system. Only TEs placed by this user will be subject to integrity update between the systems. TEs created locally on the Destination system affect this integration only in the sense that a TE from the source system with the same customID as a locally created TE cannot be transferred to the Destination side, due to the customID validation.
TEI-009	If a TE is deleted on the Destination side, and it still exists on the Source side, on the next integrity update, it will be placed again on the Destination side. However, if a TE is added on the Destination side, and it does not exist on the Source side, on the next integrity update, it will not be deleted on the Destination side. This is determined based on the user who added a TE.

4. OVERVIEW

4.1. Functional Design

The temporary elements integration is designed as a collaboration between the source and the destination adapter.

The destination adapter is a stateless adapter, which receives the payload containing the temporary elements to add or remove, validates it, and applies the valid changes to the TES of the hosting system. The destination adapter also provides the list of the ids of the existing feeders and temporary elements to the source side when queried.

The source adapter consists of several components, described in more detail in Component Design. The source adapter actions are synchronized through internal state machine, described in more detail in Source Adapter State Machine. The data is sent using a producer-consumer pattern. The producers (which can be TES subscribers described in Data Exchange, or TES readers for the integrity update, described in Integrity Update) store the data to be sent into queues which are emptied by the consumer threads, which send the data to the destination adapter. The source adapter also filters the data to be sent.

4.1.1. Data Exchange

Data exchange in normal working mode is based on a pub/sub principle. TEISourceAdapter subscribes to TES for any changes. When a change occurs, TEISourceSubscriber processes the change. This processing involves filtering the changes to check whether they pertain to the feeders of interest, and obtaining the data required for the message payload. After that, the payload objects are created and placed on the appropriate sending queue. There are four sending queues: adding cuts, deleting cuts, adding jumpers and deleting jumpers. The worker threads collect the data from the corresponding queues, form the messages for sending and send the messages.

These messages are then received by the TEIDestinationAdapter, validated, transformed in the appropriate data model and applied to TES on the destination side.

TEIDestinationAdapter returns the timestamp of the last model update on the destination side in the header of the response to every request. TEISourceAdapter compares that timestamp to the last timestamp it received from TEIDestinationAdapter. If the two differ, the integrity update process is initiated.

The sequence diagram describing the data exchange in the normal working mode is shown in Figure 4.1.

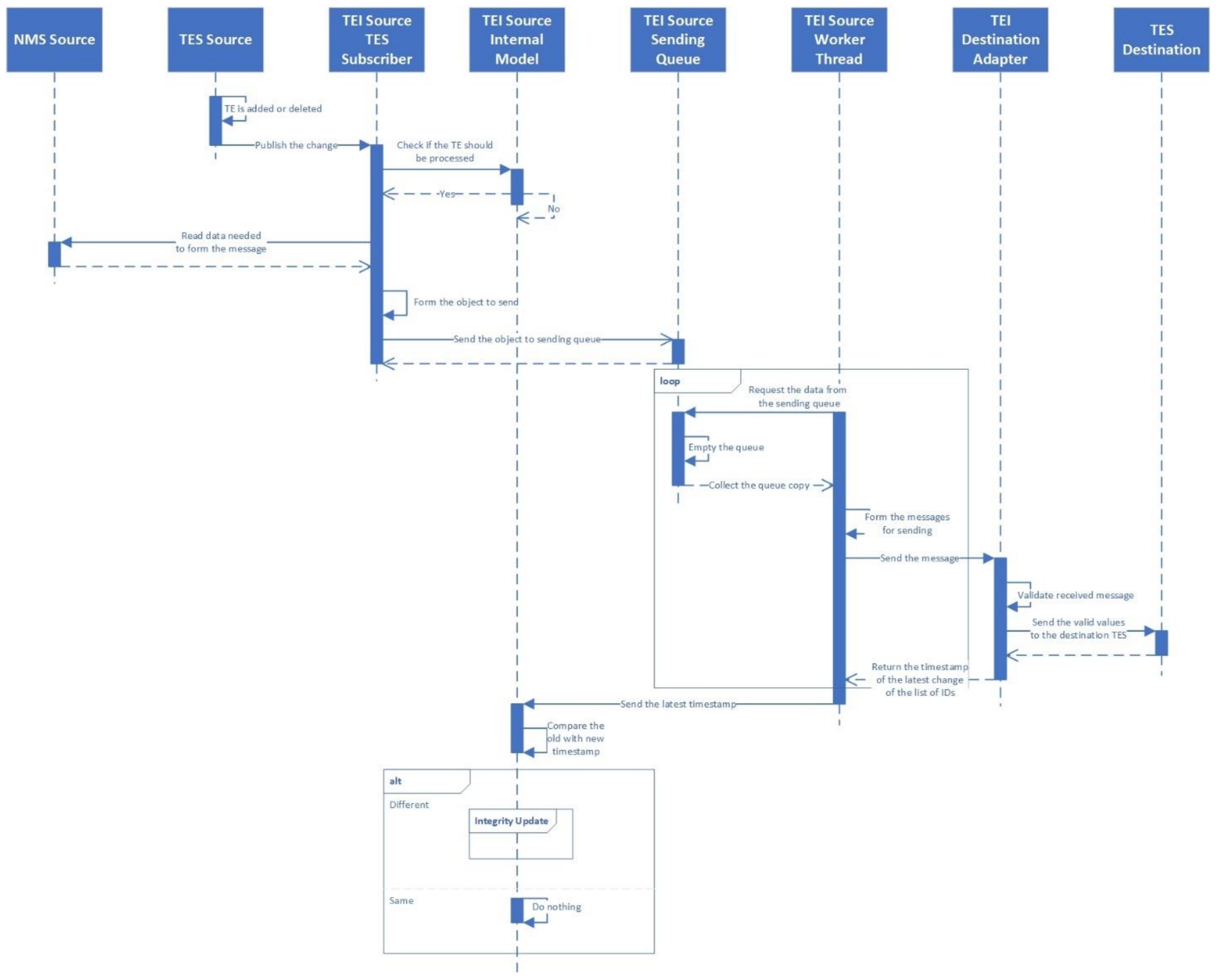
NMS Source
TES Source
TEI Source TES Subscriber
TEI Source Internal Model
TEI Source Sending Queue
TEI Source Worker Thread
TEI Destination Adapter
TES Destination
TE is added or deleted
Publish the change
Check if the TE should be processed
Yes
No
Read data needed to form the message
Form the object to send
Send the object to sending queue
loop
Request the data from the sending queue
Empty the queue
Collect the queue copy
Form the messages for sending
Send the message
Validate received message
Send the valid values to the destination TES
Return the timestamp of the latest change of the list of IDs
Send the latest timestamp
Compare the old with new timestamp
alt
Different
Integrity Update
Same
Do nothing

Figure 4.1 – Data Exchange in the normal working mode

4.1.2. Integrity Update

The integrity update process is described in the sequence diagram shown in Figure 4.2.

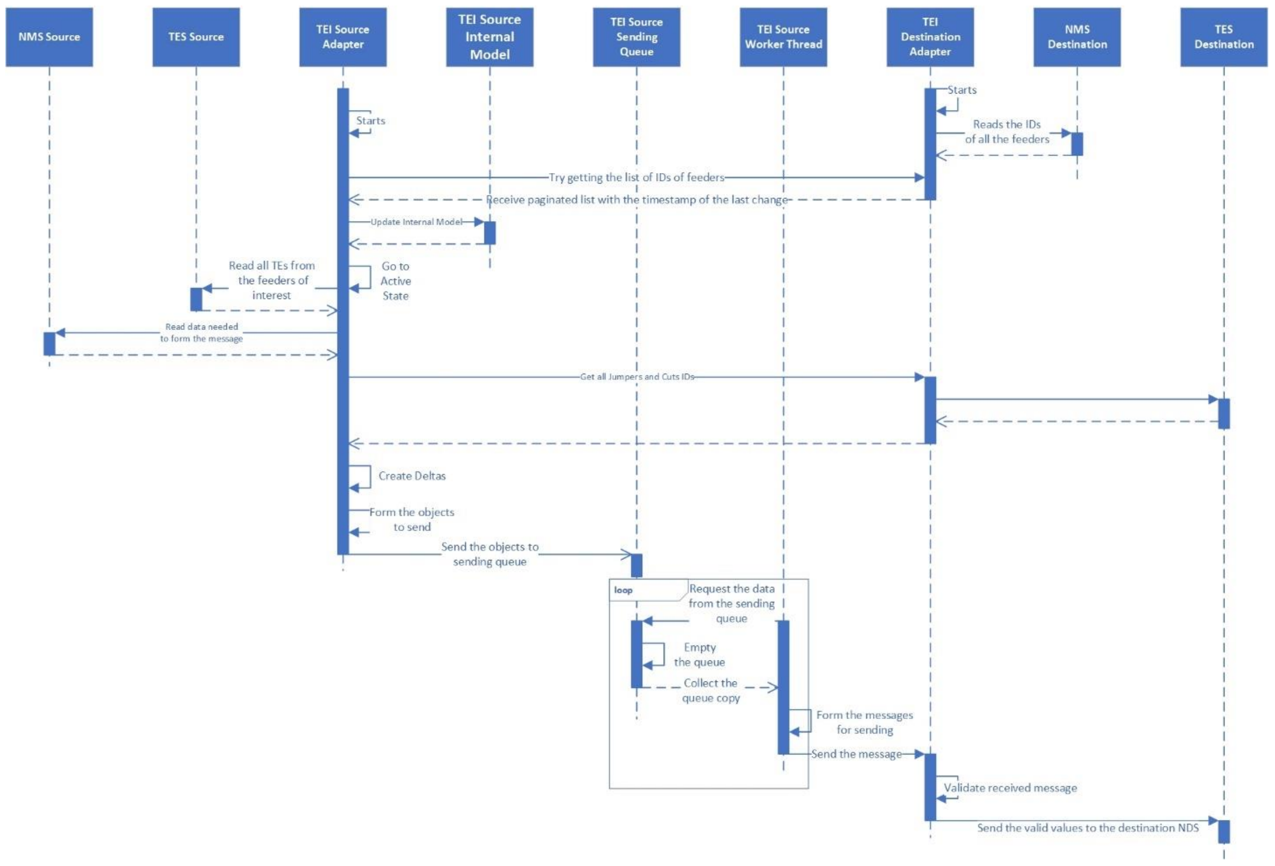
NMS Source
TES Source
TEI Source Adapter
TEI Source Internal Model
TEI Source Sending Queue
TEI Source Worker Thread
TEI Destination Adapter
NMS Destination
TES Destination
Starts
Starts
Reads the IDs of all the feeders
Try getting the list of IDs of feeders
Receive paginated list with the timestamp of the last change
Update Internal Model
Read all TEs from the feeders of interest
Go to Active State
Read data needed to form the message
Get all Jumpers and Cuts IDs
Create Deltas
Form the objects to send
Send the objects to sending queue
loop
Request the data from the sending queue
Empty the queue
Collect the queue copy
Form the messages for sending
Send the message
Validate received message
Send the valid values to the destination NDS

Figure 4.2 – Integrity Update

Integrity update process means that all existing feeders, cuts and jumpers are obtained from the destination system, then compared to the cuts and jumpers of the relevant feeders on the source system, and sending the delta to the destination system through AddCuts (Add Cuts Operation), AddJumpers (Add Jumpers Operation), DeleteCuts (Delete Cuts Operation) and DeleteJumpers (Delete Jumpers Operation) interfaces.

4.1.3. Update of the List of Relevant Feeders

The destination side dictates which TEs are to be sent by the source side. The destination side keeps the list of the feeders it contains and detects if it changed by keeping the timestamp of the last change. Only the TEs associated to the feeders from that list can be sent to the destination side. The process of notifying the source side of the change is shown in Figure 4.3.

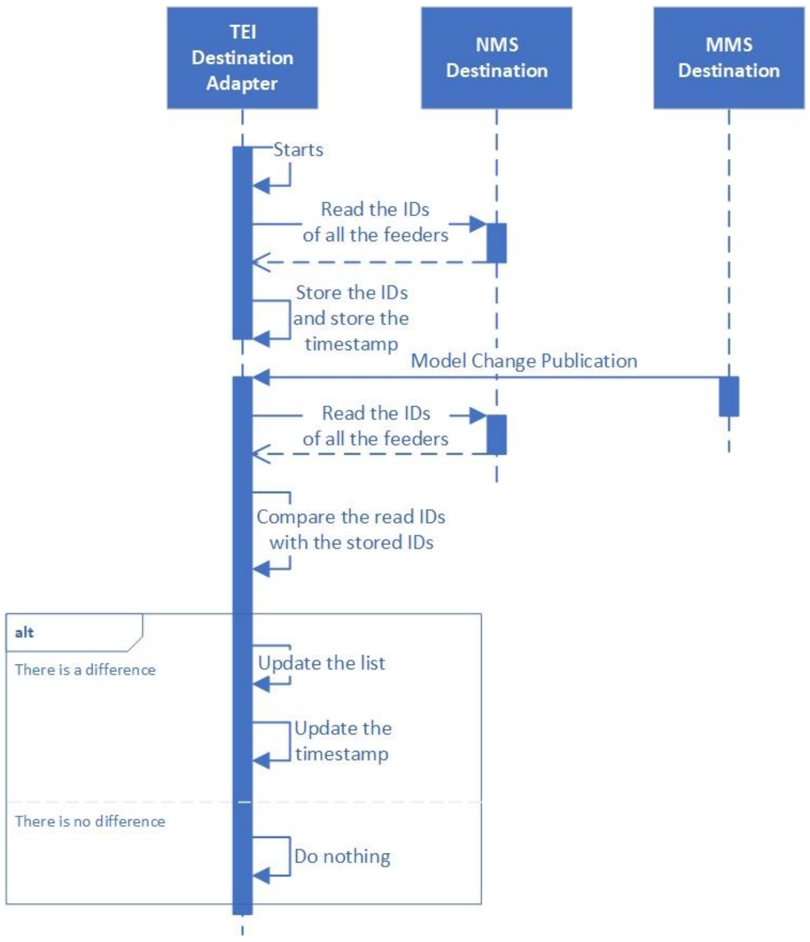

Figure 4.3 – Update the list of Feeder IDs

4.1.4. Source Adapter State Machine

The source adapter will work based on the state machine. The state machine is described in Table 4.1.

Table 4.1 – Source Adapter State Machine

Event	**State**		
	Inactive	**Active**	**Standby**
Adapter Becomes Hot	Next State: **Inactive** Actions: • Stop Worker Threads • Clear Message Queue • Clear Feeder Filter • Start the loop for GetFeederIds operation • Update Feeder Filter	Next State: **Inactive** Actions: • Stop Worker Threads • Clear Message Queue • Clear Feeder Filter • Start the loop for GetFeederIds operation • Update Feeder Filter	Next State: **Inactive** Actions: • Stop Worker Threads • Clear Message Queue • Clear Feeder Filter • Start the loop for GetFeederIds operation • Update Feeder Filter

Event	State		
	Inactive	**Active**	**Standby**
	• Issue GetFeedersIdsOK Event	• Issue GetFeedersIdsOK Event	• Issue GetFeedersIdsOK Event
Get Feeders Ids OK Event	Next State: **Active** Actions: • Start Worker Threads • Invoke GetAllCuts and GetAllJumpers • Read all cuts and jumpers from TES • CreateDelta and invoke AddCuts, AddJumpers, DeleteCuts and DeleteJumpers accordingly	Next State: **Active** Actions: • Start Worker Threads • Invoke GetAllCuts and GetAllJumpers • Read all cuts and jumpers from TES • CreateDelta and invoke AddCuts, AddJumpers, DeleteCuts and DeleteJumpers accordingly	Next State: **Standby** Actions: N/A
Failover Event	Next State: **Standby** Actions: • Stop Worker Threads	Next State: **Standby** Actions: • Stop Worker Threads	Next State: **Standby** Actions: N/A
Generate Delta Exception Event	Next State: **Inactive** Actions: • Wait 100 ms	Next State: **Inactive** Actions: • Wait 100ms	Next State: **Standby** Actions: • Wait 100ms
Sender Queue Exception Event	Next State: **Inactive** Actions: N/A	Next State: **Inactive** Actions: • Stop Worker Threads • Clear Message Queue • Clear Feeder Filter • Start the loop for GetFeederIds operation • Update Feeder Filter • Issue GetFeedersIdsOK Event	Next State: **Standby** Actions: N/A
TES Publication Received Event	Next State: **Inactive** Actions: N/A	Next State: **Active** Actions: • Process publications • Add processed messages to appropriate sender queues	Next State: **Standby** Actions: N/A

Event	**State**		
	Inactive	**Active**	**Standby**
TES Reconnected Event	Next State: **Inactive** Actions: N/A	Next State: **Active** Actions: • Create integrity update delta • Add delta messages to appropriate sender queues	Next State: **Standby** Actions: N/A

4.2. Architectural Design

This integration is designed as a pair of the adapters communicating with each other. The communication is always initiated from the source side, since the source side is the side with the higher security level. Both adapters are in their respective DMZ zones and are deployed in the hot-standby failover cluster configurations. The architecture of the solution is shown in Figure 4.4.

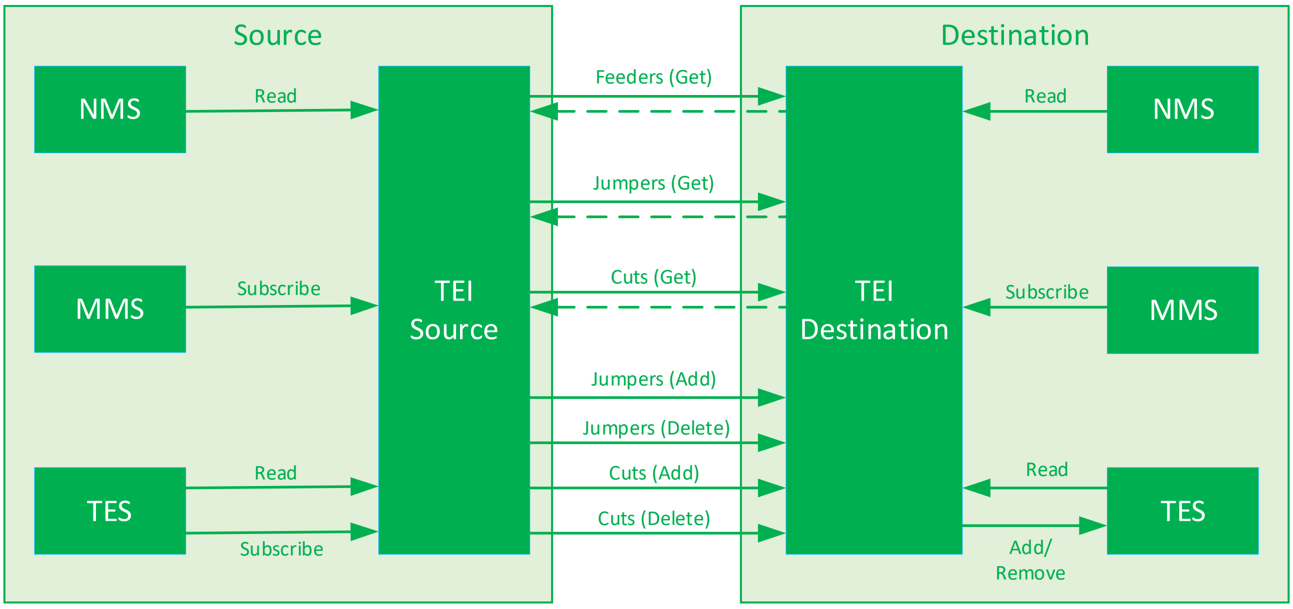

Figure 4.4 – TEI Architecture

5. DETAILED DESIGN

5.1. Component Design

The components of the source and destination adapters are described in the following chapters.

5.1.1. TEI Source Subscriber

TEI Source Subscriber is the component which subscribes to the changes in the source TES and receives the changes of the TEs. It then invokes TEIAdapterManager TEI Adapter Manager or IntegrityUpdateCoordinator (TEI Integrity Update Coordinator) which create the payload to be put on the sender queue (TEI Source Sending Queue).

5.1.2. TEI Adapter Manager

TEI Adapter Manager is the component that hosts and accesses sender queues (TEI Source Sending Queue), coordinates sender threads, and hosts and accesses TEI Source Internal Model (TEI Source Internal Model).

5.1.3. TEI Integrity Update Coordinator

TEI Integrity Update Coordinator is an orchestration component that invokes TEI TES Reader (TEI TES Reader), TEI NMS Reader (TEI NMS Reader) and also invokes GetAllCuts (Get All Cuts Operation) and GetAllJumpers (Get All Jumpers Operation) operations in order to create the appropriate integrity update delta.

5.1.4. TEI Source Internal Model

TEI Source Internal Model is the component containing the hashset of GIDs of the feeders on which there are TEs to be sent to the destination side.

5.1.5. TEI Destination Internal Model

TEI Destination Internal Model is the component containing the map of CustomIDs to GIDs of all the feeders on the destination side, based on which it is determined which TEs are eligible to be sent to the destination side. It also contains the timestamp of the last model update.

5.1.6. Destination MMS Subscriber

MMS Subscriber is the component on the destination side that is responsible for handling the model update publications. Whenever there is a model update, TEI Destination Internal model is refreshed and checked for changes.

If the model is changed (i.e. if a feeder is added or deleted), a timestamp of the last model update change is updated, which will trigger integrity update on the source side on the next call from the source side.

5.1.7. TEI NMS Reader

TEI NMS Reader is the component responsible for reading the data from NMS. This component is used both by Source and Destination adapters, but for different purposes.

On the source side, it is used to map sections and nodes customIds to their respective gids.

On the destination side, it is also used to map all the feeders customIds to their respective gids.

5.1.8. TEI TES Reader

TEI TES Reader is the component responsible for reading the data from TES. This component is used both by Source and Destination adapters, but for different purposes.

5.1.9. TEI Source Sending Queue

TEI Source Sending Queue is the component which stores the changes to be sent to the destination side. This component ensures the thread safety for this data storage.

This component also hosts the consumer thread which collects the data from the queue, transforms it in the appropriate form and sends it to the destination side using the TEI Sender Communication Component.

5.1.10. TEI Sender Communication Component

TEI Sender Communication Component is the component responsible for REST communication with the Destination adapter. This component hosts the source side APIs and encapsulates the REST clients which invoke the APIs on the destination side.

6. INTERFACES

The REST APIs used in this integration are described in the following chapters.

6.1. Add Cuts Operation

6.1.1. Overview

The addcut operation will be used to send added cuts from the source side to the destination side. This will be invoked in two use cases:

- Regular data exchange (see Data Exchange)
- Integrity update (see Integrity Update)

6.1.2. API Operation Specification

The operation specification is provided in the following table:

Table 6.1 – Add cuts – API Operation specification

Operation Specification	
Operation id	cuts
URL	/cuts
URL parameters	-
HTTP verb	PUT
Description	Method can be used to add cuts to DERMS. If successful, returns 200 OK and a payload containing the timestamp of the last feeder IDs list update, as described in Logical Format

6.1.3. API Operation Usage

Table 6.2 – Add cuts – API operation usage

API usage details	
Host system	DERMS
Consuming system	ADMS
Retry number	Needs to be configurable in adapter configuration, 3 by default
Retry condition	ADMS cannot connect to DERMS API
Time between retries	Needs to be configurable in adapter configuration, 5 seconds by default

6.1.4. Data Format

6.1.4.1. Header Format

Following attributes are used in the integration interface as part of the HTTP header:

- X-Source-ID – unique ID of the system which sent the HTTP request, supported values from the manual points integration standpoint are “ADMS” and “DERMS”. Source ID received in the HTTP request is preserved in HTTP response.
- X-Transaction-ID – needs to be populated in a form of Global Unique Identifier (GUID), unique ID of the transaction, HTTP request and HTTP response need to contain the same transaction ID.
- X-Timestamp – timestamp when request or response was created and sent.

6.1.4.2. Logical Format

The data specification of the payload in the logical format is provided in the following table:

Table 6.3 – Add cuts – logical data format

Cut		**Description**
Property Name	**Data Type**	
id	String	Unique id of the cut. This maps to custom ID of the cut.
branchId	String	Unique id of the section on which cut is placed. This maps to the custom ID of the section.
nodeId	String	Unique id of the end node of the branch closer to the cut. This maps to the custom ID of the node.
distance	Float	Distance, in percent of total length, from the beginning of the section or line busbar to the cut location
phases	Enum	Cut phases. For example, 'AB' means that phases A and B are cut, and C is continued. Applicable only for cut sections. Valid values: A, B, C, AB, AC, BC, ABC.
fieldTimestamp	DateTime	Timestamp when the element was added in the field. This is in UTC.
timestamp	DateTime	Timestamp when the item was added into the ADMS. This is in UTC.

The data specification of the response in the logical format is provided in the following table:

Table 6.4 – Add cuts – logical data format

Device Status		Description
Property Name	**Data Type**	
timestamp	DateTime	Timestamp of the last feeders IDs list update. This data is in the header of the response. This is in UTC.

6.1.4.3. Physical Format

The JSON schema for add cuts payload is shown below:

```
{
  "type": "object",
  "properties": {
    "cuts": {
      "type": "array",
      "maxItems": 100000,
      "items": {
        "type": "object",
        "properties": {
          "id": {
            "type": "string"
          },
          "branchid": {
            "type": "string"
          },
          "nodeid": {
            "type": "string"
          },
          "distance": {
            "type": "number"
          },
          "phases": {
            "type":     "string",
                  "enum": ["A", "B", "C", "AB", "AC", "BC", "ABC"]
          },
          "fieldtimestamp": {
            "type": "string",
            "format": "date-time"
          },
          "timestamp": {
            "type": "string",
            "format": "date-time"
          }
        },
        "required": [
            "id",
            "branchid",
            "nodeid",
            "distance",
            "phases",
            "fieldtimestamp",
            "timestamp"
        ]
      }
    }
  },
  "required": [
    "cuts"
  ]
}
```

Table 6.5 – Add cuts - Data specification in the physical format

Data Specification	
Name	Signals
Data content	Signal attributes as defined in Table 6.3
Expected data size	No more than 100.000 elements in the array
Data format	JSON
Compress format	No
Encoding	UTF-8 *https://en.wikipedia.org/wiki/Character_encoding*
Date time format	UTC in the ISO-8601 format

6.2. Delete Cuts Operation

6.2.1. Overview

The deletecuts operation will be used to send the ids of the existing cuts from the destination side to the source side. This will be invoked during integrity update (see Integrity Update). The source side will use those ids to identify which cuts will be sent to the destination side, and for which cuts will be deleted on the destination side.

6.2.2. API Operation Specification

The operation specification is provided in the following table:

Table 6.6 – Delete Cuts – API Operation specification

Operation Specification	
Operation id	cuts
URL	/cuts
URL parameters	-
HTTP verb	POST
Description	Method can be used to delete cuts from DERMS. If successful, returns 200 OK and a payload containing the timestamp of the last feeder IDs list update, as described in Logical Format.

6.2.3. API Operation Usage

Table 6.7 – Get All Cuts – API operation usage

API usage details	
Host system	ADMS
Consuming system	DERMS
Retry number	Needs to be configurable in adapter configuration, 3 by default
Retry condition	DERMS cannot connect to ADMS API
Time between retries	Needs to be configurable in adapter configuration, 5 seconds by default

6.2.4. Data Format

6.2.4.1. Header Format

Following attributes are used in the integration interface as part of the HTTP header:

- X-Source-ID – unique ID of the system which sent the HTTP request, supported values from the manual points integration standpoint are “ADMS” and “DERMS”. Source ID received in the HTTP request is preserved in HTTP response.
- X-Transaction-ID – needs to be populated in a form of Global Unique Identifier (GUID), unique ID of the transaction, HTTP request and HTTP response need to contain the same transaction ID.
- X-Timestamp – timestamp when request or response was created and sent.

6.2.4.2. Logical Format

The data specification in the logical format is provided in the following table:

Table 6.8 – Delete Cuts – logical data format

Device Status		Description
Property Name	Data Type	
id	String	Unique id of the cut. This maps to custom ID of the cut.

The data specification of the response in the logical format is provided in the following table:

Table 6.9 – Delete cuts – logical data format

Device Status		Description
Property Name	**Data Type**	
timestamp	DateTime	Timestamp of the last feeders IDs list update. This data is in the header of the response. This is in UTC.

6.2.4.3. *Physical Format*

The JSON schema for get signal ids payload is shown below:

```
{
	"type": "object",
	"properties":
	{
		" cutids":
		{
				"type": "array",
				"maxItems": 100000,
				"items":
			{
				"type": "string"
			},
			"required":
			[
				"cutids"
			]
		}
	}
}
```

Table 6.10 – Delete Cuts – Data specification in the physical format

Data Specification	
Name	Cutids
Data content	Cut ids as defined in Table 6.13.
Expected data size	No more than 100.000 elements in the array
Data format	JSON
Compress format	No
Encoding	UTF-8 *https://en.wikipedia.org/wiki/Character_encoding*
Date time format	No date time in the payload

6.3. Get All Cuts Operation

6.3.1. Overview

The getallcuts operation will be used to send the ids of the existing cuts from the destination side to the source side. This will be invoked during integrity update (see Integrity Update). The source side will use those ids to identify which cuts will be sent to the destination side, and for which cuts will be deleted on the destination side.

6.3.2. API Operation Specification

The operation specification is provided in the following table:

Table 6.11 – Get All Cuts – API Operation specification

Operation Specification	
Operation id	cuts
URL	/cuts
URL parameters	-
HTTP verb	GET
Description	Method will be used to provide the source ADMS side with the information on what cuts exist on the destination side. Returns the complete list of the cut ids.

6.3.3. API Operation Usage

Table 6.12 – Get All Cuts – API operation usage

API usage details	
Host system	ADMS
Consuming system	DERMS
Retry number	Needs to be configurable in adapter configuration, 3 by default
Retry condition	DERMS cannot connect to ADMS API
Time between retries	Needs to be configurable in adapter configuration, 5 seconds by default

6.3.4. Data Format

6.3.4.1. Header Format

Following attributes are used in the integration interface as part of the HTTP header:

- X-Source-ID – unique ID of the system which sent the HTTP request, supported values from the manual points integration standpoint are "ADMS" and "DERMS". Source ID received in the HTTP request is preserved in HTTP response.
- X-Transaction-ID – needs to be populated in a form of Global Unique Identifier (GUID), unique ID of the transaction, HTTP request and HTTP response need to contain the same transaction ID.
- X-Timestamp – timestamp when request or response was created and sent.

6.3.4.2. Logical Format

The data specification in the logical format is provided in the following table:

Table 6.13 – Get All Cuts – logical data format

Device Status		Description
Property Name	**Data Type**	
id	String	Unique id of the cut. This maps to custom ID of the cut

6.3.4.3. Physical Format

The JSON schema for get signal ids payload is shown below:

```
{
      "type": "object",
      "properties":
      {
            " cutids":
            {
                        "type": "array",
                        "maxItems": 100000,
                        "items":
                  {
                        "type": "string"
                  },
                  "required":
                  [
                        "cutids"
                  ]
            }
      }
}
```

Table 6.14 – Get All Cuts – Data specification in the physical format

Data Specification	
Name	Cutids
Data content	Cut ids as defined in Table 6.13
Expected data size	No more than 100.000 elements in the array

Data Specification	
Data format	JSON
Compress format	No
Encoding	UTF-8 *https://en.wikipedia.org/wiki/Character_encoding*
Date time format	No date time in the payload

6.4. Add Jumpers Operation

6.4.1. Overview

The addjumper operation will be used to send added jumpers from the source side to the destination side. This will be invoked in two use cases:

- Regular data exchange (see Data Exchange)
- Integrity update (see Integrity Update)

6.4.2. API Operation Specification

The operation specification is provided in the following table:

Table 6.15 – Add jumpers – API Operation specification

Operation Specification	
Operation id	jumpers
URL	/jumpers
URL parameters	-
HTTP verb	PUT
Description	Method can be used to add jumpers to DERMS. If successful, returns 200 OK and a payload containing the timestamp of the last feeder IDs list update, as described in Logical Format.

6.4.3. API Operation Usage

Table 6.16 – Add jumpers – API operation usage

API usage details	
Host system	DERMS
Consuming system	ADMS
Retry number	Needs to be configurable in adapter configuration, 3 by default

API usage details	
Retry condition	ADMS cannot connect to DERMS API
Time between retries	Needs to be configurable in adapter configuration, 5 seconds by default

6.4.4. Data Format

6.4.4.1. Header Format

Following attributes are used in the integration interface as part of the HTTP header:

- X-Source-ID – unique ID of the system which sent the HTTP request, supported values from the manual points integration standpoint are “ADMS” and “DERMS”. Source ID received in the HTTP request is preserved in HTTP response.
- X-Transaction-ID – needs to be populated in a form of Global Unique Identifier (GUID), unique ID of the transaction, HTTP request and HTTP response need to contain the same transaction ID.
- X-Timestamp – timestamp when request or response was created and sent.

6.4.4.2. Logical Format

The data specification of the payload in the logical format is provided in the following table:

Table 6.17 – Add jumpers – logical data format

Jumper		Description
Property Name	Data Type	
id	String	Unique id of the jumper. This maps to custom ID of the jumper.
fieldTimestamp	DateTime	Timestamp when the element was added in the field. This is in UTC.
timestamp	DateTime	Timestamp when the item was added into the ADMS. This is in UTC.
length	Float	Total length of the jumper [m]
point1	Point	First point of a jumper, as defined in Table 6.18.
point2	Point	First point of a jumper, as defined in Table 6.18.

Table 6.18 – Add jumpers point – logical data format

Point		Description
Property Name	Data Type	
nodeid	String	Id of the node at that end of the connector
sectionid	String	Id of the section to which that end of the jumper is attached. May be NULL if the jumper is attached to the node on that end.

Point		**Description**
Property Name	**Data Type**	
distance	Float	Distance from the beginning of the section to the point of connection, in percent of its total length.
phase	Enum	Phases at the first end of the connector. The other end must have the same number of phases. Valid values: A, B, C, AB, AC, BC, ABC.

The data specification of the response in the logical format is provided in the following table:

Table 6.19 – Add jumpers – logical data format

Device Status		**Description**
Property Name	**Data Type**	
timestamp	DateTime	Timestamp of the last feeders IDs list update. This data is in the header of the response. This is in UTC.

6.4.4.3. Physical Format

The JSON schema for add jumpers payload is shown below:

```
{
  "type": "object",
  "properties": {
    "jumpers": {
      "type": "array",
      "maxItems": 100000,
      "items": {
        "type": "object",
        "properties": {
          "id": {
            "type": "string"
          },
          "length": {
            "type": "number"
          },
          "point1": {
            "type": "object",
                  "properties": {
                        "nodeid": {
                              "type": "string"
                        },
                        "sectionid": {
                              "type": "string"
                        },
                        "distance": {
                              "type": "number"
                        },
                        "phase": {
                        "type":     "string",
                        "enum": ["A", "B", "C", "AB", "AC", "BC", "ABC"]
                        }
                  },
                  "required": [
                        "distance",
                        "phase"
                        ]
            },
   "point2": {
            "type": "object",
                  "properties": {
                        "nodeid": {
                              "type": "string"
                        },
                        "sectionid": {
                              "type": "string"
                        },
                        "distance": {
                              "type": "number"
                        },
                        "phase": {
                              "type":     "string",
                              "enum": ["A", "B", "C", "AB", "AC", "BC", "ABC"]
                        }
                  },
                  "required": [
                        "distance",
                        "phase"
                        ]
            },
          "fieldtimestamp": {
            "type": "string",
            "format": "date-time"
          },
          "timestamp": {
            "type": "string",
            "format": "date-time"
          }
        },
        "required": [
                  "id",
            "length",
                  "point1",
                  "point2",
            "fieldtimestamp",
            "timestamp"
        ]
      }
    }
  },
  "required": [
    "jumpers"
  ]
}
```

Table 6.20 – Add jumpers - Data specification in the physical format

Data Specification	
Name	Signals
Data content	Signal attributes as defined in Table 6.3
Expected data size	No more than 100.000 elements in the array
Data format	JSON
Compress format	No
Encoding	UTF-8 *https://en.wikipedia.org/wiki/Character_encoding*
Date time format	UTC in the ISO-8601 format

6.5. Delete Jumpers Operation

6.5.1. Overview

The deletejumpers operation will be used to send the ids of the existing jumpers from the destination side to the source side. This will be invoked during integrity update (see Integrity Update). The source side will use those ids to identify which jumpers will be sent to the destination side, and for which jumpers will be deleted on the destination side.

6.5.2. API Operation Specification

The operation specification is provided in the following table:

Operation Specification	
Operation id	jumpers
URL	/jumpers
URL parameters	-
HTTP verb	POST
Description	Method can be used to delete jumpers from DERMS. If successful, returns 200 OK and a payload containing the timestamp of the last feeder IDs list update, as described in Logical Format.

6.5.3. API Operation Usage

Table 6.21 – Get All Jumpers – API operation usage

API usage details	
Host system	ADMS

API usage details	
Consuming system	DERMS
Retry number	Needs to be configurable in adapter configuration, 3 by default
Retry condition	DERMS cannot connect to ADMS API
Time between retries	Needs to be configurable in adapter configuration, 5 seconds by default

6.5.4. Data Format

6.5.4.1. Header Format

Following attributes are used in the integration interface as part of the HTTP header:

- X-Source-ID – unique ID of the system which sent the HTTP request, supported values from the manual points integration standpoint are “ADMS” and “DERMS”. Source ID received in the HTTP request is preserved in HTTP response.
- X-Transaction-ID – needs to be populated in a form of Global Unique Identifier (GUID), unique ID of the transaction, HTTP request and HTTP response need to contain the same transaction ID.
- X-Timestamp – timestamp when request or response was created and sent.

6.5.4.2. Logical Format

The data specification in the logical format is provided in the following table:

Table 6.22 – Delete Jumpers – logical data format

Device Status		Description
Property Name	Data Type	
id	String	Unique id of the jumper. This maps to custom ID of the jumper

The data specification of the response in the logical format is provided in the following table:

Table 6.23 – Delete jumpers – logical data format

Device Status		Description
Property Name	Data Type	
timestamp	DateTime	Timestamp of the last feeders IDs list update. This data is in the header of the response. This is in UTC.

6.5.4.3. Physical Format

The JSON schema for get signal ids payload is shown below:

```
{
        "type": "object",
        "properties":
        {
                " jumperids":
                {
                        "type": "array",
                        "maxItems": 100000,
                        "items":
                        {
                                "type": "string"
                        },
                        "required":
                        [
                                "jumperids"
                        ]
                }
        }
}
```

Table 6.24 – Delete Jumpers – Data specification in the physical format

Data Specification	
Name	Jumperids
Data content	Jumper ids as defined in Table 6.13
Expected data size	No more than 100.000 elements in the array
Data format	JSON
Compress format	No
Encoding	UTF-8 https://en.wikipedia.org/wiki/Character_encoding
Date time format	No date time in the payload

6.6. Get All Jumpers Operation

6.6.1. Overview

The getalljumpers operation will be used to send the ids of the existing jumpers from the destination side to the source side. This will be invoked during integrity update (see Integrity Update). The source side will use those ids to identify which jumpers will be sent to the destination side, and for which jumpers will be deleted on the destination side.

6.6.2. API Operation Specification

The operation specification is provided in the following table:

Table 6.25 – Get All Jumpers – API Operation specification

Operation Specification	
Operation id	jumpers

Operation Specification	
URL	/jumpers
URL parameters	-
HTTP verb	GET
Description	Method will be used to provide the source ADMS side with the information on what jumpers exist on the destination side. Returns the complete list of the jumper ids.

6.6.3. API Operation Usage

Table 6.26 – Get All Jumpers – API operation usage

API usage details	
Host system	ADMS
Consuming system	DERMS
Retry number	Needs to be configurable in adapter configuration, 3 by default
Retry condition	DERMS cannot connect to ADMS API
Time between retries	Needs to be configurable in adapter configuration, 5 seconds by default

6.6.4. Data Format

6.6.4.1. Header Format

Following attributes are used in the integration interface as part of the HTTP header:

- X-Source-ID – unique ID of the system which sent the HTTP request, supported values from the manual points integration standpoint are “ADMS” and “DERMS”. Source ID received in the HTTP request is preserved in HTTP response.
- X-Transaction-ID – needs to be populated in a form of Global Unique Identifier (GUID), unique ID of the transaction, HTTP request and HTTP response need to contain the same transaction ID.
- X-Timestamp – timestamp when request or response was created and sent.

6.6.4.2. Logical Format

The data specification in the logical format is provided in the following table:

Table 6.27 – Get All Jumpers – logical data format

Device Status		Description
Property Name	**Data Type**	
id	String	Unique id of the jumper. This maps to custom ID of the jumper

6.6.4.3. *Physical Format*

The JSON schema for get signal ids payload is shown below:

```
{
      "type": "object",
      "properties":
      {
            " jumperids":
            {
                        "type": "array",
                        "maxItems": 100000,
                        "items":
                  {
                        "type": "string"
                  },
                  "required":
                  [
                        "jumperids"
                  ]
            }
      }
}
```

6.7. Get All Feeders Operation

6.7.1. Overview

The getallfeeders operation will be used to send the ids of the existing feeders from the destination side to the source side. This will be invoked during integrity update (see Integrity Update). The source side will use those ids to identify which feeders will be sent to the destination side, and for which feeders will be deleted on the destination side.

6.7.2. API Operation Specification

The operation specification is provided in the following table:

Table 6.28 – Get All Feeders – API Operation specification

Operation Specification	
Operation id	feeders
URL	/feeders

Operation Specification	
URL parameters	-
HTTP verb	GET
Description	Method will be used to provide the source ADMS side with the information on what feeders exist on the destination side. Returns the complete list of the feeder ids.

6.7.3. API Operation Usage

Table 6.29 – Get All Feeders – API operation usage

API usage details	
Host system	ADMS
Consuming system	DERMS
Retry number	Needs to be configurable in adapter configuration, 3 by default
Retry condition	DERMS cannot connect to ADMS API
Time between retries	Needs to be configurable in adapter configuration, 5 seconds by default

6.7.4. Data Format

6.7.4.1. Header Format

Following attributes are used in the integration interface as part of the HTTP header:

- X-Source-ID – unique ID of the system which sent the HTTP request, supported values from the manual points integration standpoint are “ADMS” and “DERMS”. Source ID received in the HTTP request is preserved in HTTP response.
- X-Transaction-ID – needs to be populated in a form of Global Unique Identifier (GUID), unique ID of the transaction, HTTP request and HTTP response need to contain the same transaction ID.
- X-Timestamp – timestamp when request or response was created and sent.

6.7.4.2. Logical Format

The data specification in the logical format is provided in the following table:

Table 6.30 – Get All Feeders – logical data format

Device Status		Description
Property Name	Data Type	
id	String	Unique id of the feeder. This maps to custom ID of the feeder

6.7.4.3. *Physical Format*

The JSON schema for get signal ids payload is shown below:

```
{
      "type": "object",
      "properties":
      {
            "feederids":
            {
                        "type": "array",
                        "maxItems": 100000,
                        "items":
                  {
                        "type": "string"
                  },
                  "required":
                  [
                        "feederids"
                  ]
            }
      }
}
```

7. ADDITIONAL CONSIDERATIONS

7.1. Quality and Testing

The testing of this integration can prove to be challenging, because two ADMS systems are needed to perform end to end testing properly. Each adapter can be tested separately.

Destination adapter can be tested with any REST test client.

For the source adapter, there should exist a test service that will mimic the destination side.

This test tool should:

- Read a subset of feeders from the source side and store that set in the internal cache.
- Expand and contract that subset in the internal cache, mimicking the model update.
- Write the received payload on the console or in some report for the examination of the test results.
- Send non-existing ids of cuts and jumpers along with the existing cuts and jumpers on get operations for cuts and jumpers, so that integrity update functionality can be tested.

The test cases hereafter will assume the two systems are set up, but the same can be achieved with test clients / test services.

7.1.1. Test GetFeederIds Interface

The purpose of the test is to demonstrate the blue sky scenario for the GetFeederIds interface.

7.1.1.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.1.2. Execution

1. Start the Integration service on the destination side.
2. TEIDestinationAdapter should start and switch to Hot state together with the Integration Service. This can also be verified in the adapter logs.
3. The SensitiveLog of the TEIDestinationAdapter should contain the list of all relevant feeders and the timestamp when it was loaded.
4. Invoke GET operation on feeders resource, as defined in Get All Feeders Operation (example: *http://localhost:8082/TeiDestinationService/feeder*).
5. The payload of the response should contain the list of ids of all feeders on the system. The payload should conform to the format described in Data Format. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp of when the adapter was started (can be found in sensitive log).
6. The adapter log and the adapter sensitive log should follow all the actions.

7.1.2. Test GetCuts Interface

The purpose of the test is to demonstrate the blue sky scenario for the GetCuts interface.

7.1.2.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.2.2. Execution

1. Start the Integration service on the destination side.
2. TEIDestinationAdapter should start and switch to Hot state together with the Integration Service. This can also be verified in the adapter logs.
3. Invoke GET operation on cuts resource, as defined in Get All Cuts Operation (example: *http://localhost:8082/TeiDestinationService/cuts*).
4. The payload of the response should contain the list of ids of all cuts on the system with the user "External System". The other temporary elements should not be included in the response. The response status should be 200 OK. The response should be as defined in Data Format.
5. The adapter log and the adapter sensitive log should follow all the actions.

7.1.3. Test GetJumpers Interface

The purpose of the test is to demonstrate the blue sky scenario for the GetJumpers interface.

7.1.3.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.3.2. Execution

1. Start the Integration service on the destination side.
2. TEIDestinationAdapter should start and switch to Hot state together with the Integration Service. This can also be verified in the adapter logs.
3. Invoke GET operation on jumpers resource, as defined in Get All Jumpers Operation (example: *http://localhost:8082/TeiDestinationService/jumpers*).
4. The payload of the response should contain the list of ids of all jumpers on the system with the user "External System". The other temporary elements should not be included in the response. The response status should be 200 OK. The response should be as defined in Data Format.
5. The adapter log and the adapter sensitive log should follow all the actions.

7.1.4. Test AddCuts Interface

The purpose of the test is to demonstrate the blue sky scenario for the AddCuts interface.

7.1.4.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.4.2. Execution

1. Start the Integration service on the destination side.
2. TEIDestinationAdapter should start and switch to Hot state together with the Integration Service. This can also be verified in the adapter logs.

3. Invoke PUT operation on cuts resource, as defined in Add Cuts Operation (example: *http://localhost:8082/TeiDestinationService/cuts*).
4. The payload of the request should contain the json represenation of the list of the added cuts, as defined in Data Format.
5. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp of the starting the adapter. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.
6. The cuts should be placed on the destination system, as specified in the message. The user of the placed cuts should be "External System".
7. Invoke PUT operation on cuts resource, as defined in Add Cuts Operation (example: *http://localhost:8082/TeiDestinationService/cuts*). The payload of the request should contain the json represenation of the list of four cuts, as defined in the design document. The first cut should have an ID of an already existing cut in the system. The second cut should contain a branch that does not exist in the system. The third cut should contain a node that does not exist in the system. The fourth cut should be valid, as in previous step.
8. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp of the starting the adapter. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.
9. None of the first three cuts should be placed on a system. The fourth cut should be placed on the destination system, as in the previous step.
10. The adapter log and the adapter sensitive log should follow all the actions.

7.1.5. Test DeleteCuts Interface

The purpose of the test is to demonstrate the blue sky scenario for the DeleteCuts interface.

7.1.5.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.5.2. Execution

1. Start the Integration service on the destination side.
2. TEIDestinationAdapter should start and switch to Hot state together with the Integration Service. This can also be verified in the adapter logs.
3. Invoke POST operation on cuts resource, as defined in Delete Cuts Operation (example: *http://localhost:8082/TeiDestinationService/cuts*).
4. The payload of the request should contain the json represenation of the list of the ids of cuts to be deleted, as defined in Data Format.
5. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp of the starting the adapter. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.
6. The cuts with IDs specified in the list should be deleted from the system.
7. The adapter log and the adapter sensitive log should follow all the actions.

7.1.6. Test AddJumpers Interface

The purpose of the test is to demonstrate the blue sky scenario for the AddJumpers interface.

7.1.6.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.6.2. Execution

1. Start the Integration service on the destination side.
2. TEIDestinationAdapter should start and switch to Hot state together with the Integration Service. This can also be verified in the adapter logs.
3. Invoke PUT operation on jumpers resource, as defined in Add Jumpers Operation (example: *http://localhost:8082/TeiDestinationService/jumpers*).
4. The payload of the request should contain the json represenation of the list of the added jumpers, as defined in Data Format.
5. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp of the starting the adapter. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.
6. The jumpers should be placed on the destination system, as specified in the message. The user of the placed jumpers should be "External System".
7. Invoke PUT operation on jumpers resource, as defined in Add Jumpers Operation. (example: *http://localhost:8082/TeiDestinationService/jumpers*). The payload of the request should contain the json represenation of the list of four jumpers, as defined in the design document. The first jumper should have an ID of an already existing jumper in the system. The second jumper should contain a branch that does not exist in the system. The third jumper should contain a node that does not exist in the system. The fourth jumper should be valid, as in previous step.
8. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp of the starting the adapter. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.
9. None of the first three jumpers should be placed on a system. The fourth jumper should be placed on the destination system, as in the previous step.
10. The adapter log and the adapter sensitive log should follow all the actions.

7.1.7. Test DeleteJumpers Interface

The purpose of the test is to demonstrate the blue sky scenario for the DeleteJumpers interface.

7.1.7.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.7.2. Execution

1. Start the Integration service on the destination side.
2. TEIDestinationAdapter should start and switch to Hot state together with the Integration Service. This can also be verified in the adapter logs.

3. Invoke POST operation on jumpers resource, as defined in Delete Jumpers Operation (example: *http://localhost:8082/TeiDestinationService/jumpers*).
4. The payload of the request should contain the json representation of the list of the ids of jumpers to be deleted, as defined in Data Format.
5. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp of the starting the adapter. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.
6. The jumpers with IDs specified in the list should be deleted from the system.
7. The adapter log and the adapter sensitive log should follow all the actions.

7.1.8. Test Model Update on Destination

The purpose of the test is to demonstrate the response of the adapter to the model update on the system.

7.1.8.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.8.2. Execution

1. Start the Integration service on the destination side.
2. TEIDestinationAdapter should start and switch to Hot state together with the Integration Service. This can also be verified in the adapter logs.
3. Invoke any operation from the previous test cases.
4. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp of the starting the adapter. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.
5. Perform a model update on the system which does not include adding or removing feeders.
6. Invoke any operation from the previous test cases.
7. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp which is the same as the previous time. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.
8. Perform a model update on the system which includes adding or removing a feeder.
9. Invoke any operation from the previous test cases.
10. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp which is newer than the previous time. The timestamp should be the one of the model update. It can be deduced from the logs. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.

7.1.9. Test Source Adapter Startup Without Destination Side

The purpose of the test is to demonstrate initialization of the TEISourceAdapter.

7.1.9.1. Preconditions

DMSRealTime service should not be running on the destination side.

7.1.9.2. Execution

1. Start the Integration service on the source side.
2. TEIDestinationAdapter should start and switch to Hot state together with the Integration Service.
3. The logs should indicate the following (in this order, but there could be some other logs in between them):
    a. TEIAdapterManager - Initializing...
    b. [StateMachine]: AdapterBecomesHotEvent event in Inactive State.
    c. [StateMachine]: AdapterBecomesHotEvent switching to Inactive State.
    d. The remote server returned an error: (503) Server Unavailable. (this line is the last one and repeats every 5s or so, depending on the retry timeout setting of GetFeedersIds client) (this error can also be any 500 error).

7.1.10. Test Source Adapter Startup With Destination Side

This test case demonstrates initialization of the TEISourceAdapter and its connection to the Destination side. There should either be another system with TEIDestinationAdapter on it, or the test service should run.

7.1.10.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.10.2. Execution

1. Start the Integration service on the source side. The Integration service on the destination side should not run at this point.
2. TEISourceAdapter should start and switch to Hot state together with the Integration Service.
3. The logs should indicate the following (in this order, but there could be some other logs in between them):
    a. TEIAdapterManager - Initializing...
    b. [StateMachine]: AdapterBecomesHotEvent event in Inactive State.
    c. [StateMachine]: AdapterBecomesHotEvent switching to Inactive State.
    d. The remote server returned an error: (503) Server Unavailable. (this line is the last one and repeats every 5s or so, depending on the retry timeout setting of GetFeedersIds client) (this error can also be any 500 error).
4. Start the destination side.
5. The logs should indicate the following (in this order, but there could be some other logs in between them):
    a. [StateMachine]: GetFeedersIdsOKEvent in Inactive State
    b. [StateMachine]: GetFeedersIdsOKEvent - Switching to ActiveState State
    c. TemporaryElementReader - ReadAllCuts - Enter
    d. TemporaryElementReader - ReadAllJumpers - Enter
6. Check the destination side. The requested feeders from the destination side should be shown in the test app or in the sensitive logs. All TEs that do not exist on the destination side should be sent to the destination side. In case the test app is used, three cuts and three jumpers with DeleteXXX

custom ID (XXX - Cut1, Cut2, Cut3...) should be in the delete cuts and delete jumpers payload delivered. These are the hardcoded IDs of the TEs in the test app that simulate TEs that exist on the destination side and not on the source side. Of course, if the TE with that customID exists on the source side, this customID should not be in the delete payload.

7.1.11. Test Sending and Deleting Cuts

This test case demonstrates sending cuts to the destination side.

7.1.11.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.11.2. Execution

1. Both Source and Destination side should be up and running.
2. On the source side, place a cut on one of the feeders listed on the destination side.
3. The cut should appear on the destination side, as placed on the source side. In case a test app is used, the destination test service should indicate that the cut was received, with the payload matching the data entered in DMD. The payload should be as defined in Data Format.
4. On the source side, remove a cut from one of the feeders listed on the destination side.
5. The corresponding cut should disappear on the destination side. In case a test app is used, the destination test service should indicate that the cut was received, with the payload matching the data entered in DMD. The payload should be as defined in Data Format.
6. On the source side, place a cut on one of the feeders not listed on the destination side.
7. There should be no communication with the destination adapter or destination test service.
8. On the source side, remove a cut from one of the feeders not listed on the destination side.
9. There should be no communication with the destination adapter or destination test service.

7.1.12. Test Sending and Deleting Jumpers

This test case demonstrates sending jumpers to the destination side.

7.1.12.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.12.2. Execution

1. Both Source and Destination side should be up and running.
2. On the source side, place a jumper on one of the feeders listed on the destination side.
3. The jumper should appear on the destination side, as placed on the source side. In case a test app is used, the destination test service should indicate that the jumper was received, with the payload matching the data entered in DMD. The payload should be as defined in Data Format.
4. On the source side, remove a jumper from one of the feeders listed on the destination side.
5. The corresponding jumper should disappear on the destination side. In case a test app is used, the destination test service should indicate that the jumper was received, with the payload matching the data entered in DMD. The payload should be as defined in Data Format.

6. On the source side, place a jumper on one of the feeders not listed on the destination side.
7. There should be no communication with the destination adapter or destination test service.
8. On the source side, remove a jumper from one of the feeders not listed on the destination side.
9. There should be no communication with the destination adapter or destination test service.

7.1.13. Reconnect the Source Side After the Failure of the Destination Side

This test case demonstrates how the TEISourceAdapter reconnects to the Destination side after the Destination side fails and restarts.

7.1.13.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.13.2. Execution

1. Both Source and Destination side should be up and running.
2. On the source side, place a cut or jumper on one of the feeders listed on the destination side.
3. Everything should be like in one of the previous two test cases.
4. Shut down the destination side.
5. There should be no changes in the logs or behavior of the source side.
6. On the source side, place a cut or jumper on one of the feeders listed on the destination side.
7. In the logs there should be an indication that the destination side is unavailable, indicating also that the state machine has transited to Inactive state. The logs should indicate the:
    a. The remote server returned an error: (503) Server Unavailable. (this line is the last one and repeats every 5s or so, depending on the retry timeout setting of GetFeedersIds client) (this error can also be any 500 error).
8. Start the destination side.
9. In the logs, there should be an indication that the integrity update was performed and that the state machine has transited to Active state. Destination side should be aligned with the Source side after that.

7.1.14. Test Reaction of the Source Adapter to Model Update on the Destination Side

The purpose of the test is to demonstrate the response of the source adapter to the model update on the destination system.

7.1.14.1. Preconditions

DMSRealTime service must be started and in Hot State on the destination side.

7.1.14.2. Execution

1. Start the Integration service on both source and destination side.
2. Integrity update should be performed and systems aligned, as described in Test Source Adapter Startup With Destination Side.

3. Perform a model update on the destination system which does not include adding or removing feeders.
4. Place a cut or jumper on the source side that does not affect the feeders of interest.
5. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp which is the same as the previous time. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.
6. Perform a model update on the system which includes adding or removing a feeder.
7. There should be no changes in the logs or behavior of the source side.
8. Place a cut or jumper on the source side that does not affect the feeders of interest.
9. The response status should be 200 OK. The response should contain a header LastModelUpdateTimestamp with the timestamp which is newer than the previous time. The timestamp should be the one of the model update. It can be deduced from the logs. The format of the timestamp should be yyyy-MM-ddTHH:mm:ss.fff.
10. The cut or the jumper should be shown on the destination side. Right after that, an integrity update should be performed, as described in Test Source Adapter Startup With Destination Side.

7.2. Deployment and Configuration

7.2.1. Deployment

The adapters are installed as part of the Integration service. Each adapter is installed as part of its own add-on. These add-ons need to be selected on the corresponding system: TEISource add-on should be selected on the source system, whereas TEIDestination should be selected on the destination system. This will deploy the binaries and configuration files to the server hosting the Integration service.

After the add-on installation, Config Tool needs to be run.

7.2.2. Configuration

TEIDestinationAdapter and TEISourceAdapter have two configuration files each:

- Registry configuration file
- WebServiceConfig configuration file

The parameters of interest are described in the tables below. The parameters that are not described are standard parameters used in all adapters, as described in *EcoStruxure GridOps Management Suite 3.10 Enterprise Integration Platform - Functional Specification* document [1].

Table 7.1 – TEI Source WebService GetFeederIds Configuration

Parameter	Description
ServiceUrl	URL of the GetFeederIds endpoint.
NumOfRetries	Number of retries if the call fails. For this parameter, this value must be -1, as this call needs to be repeated until it succeeds.
RetryTimeout	Timeout between retries. Default value: 00:00:05.

Parameter	Description
AuthenticationType	Possible values for AuthenticationType are: None, OneWaySslAuthentication, TwoWaySslAuthentication, TransportWithMessageCredentialAuthentication, MessageSecurityCertificate.
ClientCertificateName	Name of the certificate to be used. Mandatory if AuthenticationType is anything other than None.

Table 7.2 – TEISource WebService Configuration for Cuts

Parameter	Description
ServiceUrl	Base URL of the cuts endpoint.
NumOfRetries	Number of retries if the call fails. After this number is exceeded, communication exception is thrown which will initiate the transition of MPISourceAdapter state machine to the inactive state. Default value is 5. Has to be positive value.
RetryTimeout	Timeout between retries. Default value: 00:00:05.
AuthenticationType	Possible values for AuthenticationType are: None, OneWaySslAuthentication, TwoWaySslAuthentication, TransportWithMessageCredentialAuthentication, MessageSecurityCertificate.
ClientCertificateName	Name of the certificate to be used. Mandatory if AuthenticationType is anything other than None.

Table 7.3 – TEISource WebService Configuration for Jumpers

Parameter	Description
ServiceUrl	BaseURL of the jumpers endpoint.
NumOfRetries	Number of retries if the call fails. After this number is exceeded, communication exception is thrown which will initiate the transition of MPISourceAdapter state machine to the inactive state. Default value is 5. Has to be positive value.
RetryTimeout	Timeout between retries. Default value: 00:00:05.
AuthenticationType	Possible values for AuthenticationType are: None, OneWaySslAuthentication, TwoWaySslAuthentication, TransportWithMessageCredentialAuthentication, MessageSecurityCertificate.
ClientCertificateName	Name of the certificate to be used. Mandatory if AuthenticationType is anything other than None.

Table 7.4 – MPI Destination WebService Configuration

Parameter	Description
ServiceUrl	Base URL of the destination endpoints. This value will be used to form the URLs for the getfeederids, jumpers and cuts endpoints.
AuthenticationType	Possible values for AuthenticationType are: None, OneWaySslAuthentication, TwoWaySslAuthentication, TransportWithMessageCredentialAuthentication, MessageSecurityCertificate.
ServiceCertificateName	Name of the certificate to be used. Mandatory if AuthenticationType is anything other than None.

7.3. Upgradability

The upgrade of the EcoStruxure ADMS version should have minimal effect on this integration. The interfaces used by these adapters are:

- GDA Queries
- ITemporaryElementsManagementContract
- GetFeederIds (see Get All Feeders Operation)
- Cuts (see Add Cuts Operation, Delete Cuts Operation, and Get All Cuts Operation)
- Jumpers (see Add Jumpers Operation, Delete Jumpers Operation, and Get All Jumpers Operation)

It is unlikely that these interfaces will change. Moreover, if they do not change, the adapters can be used to integrate two ADMS systems on different versions.

7.4. Extensibility

No extensibility is planned or intended for this integration.

7.5. Security

Security aspects of this integration are in accordance with Enterprise Integration Platform, as described in *EcoStruxure GridOps Management Suite 3.10 Enterprise Integration Platform - Functional Specification* document [1].

8. DEFINITIONS AND ABBREVIATIONS

Definition/Abbreviation	Description
ADMS	Advanced Distribution Management System
API	Application Programming Interface
DERMS	Distributed Energy Resources Management Systems
DMD	Dynamic Mimic Diagram
DMZ	Demilitarized Zone
GDA	Generic Data Access
GUID	Global Unique Identifier
HTTP	Hyper-Text Transfer Protocol
ID	Identifier
ISO	International Organization for Standardization
JSON	JavaScript Object Notation – open standard file format and data interchange format
MMS	Model Management Service
NMS	Network Model Service
REST	Representational State Transfer
TE	Temporary Element
TEI	Temporary Elements Integration
TES	Temporary Elements Service
URL	Uniform Resource Locator