Meter Data Inbound API

***Overview***

This API specification was designed to explain the process for transferring Meter Data information into nMarket from a participant’s external systems. A market participant can define the types of meter read  sources for which they will be transferring data into nMarket. One version for source will be stored in nMarket for a location and interval.

**Supported Markets**

* MISO

***Process***

The nMarket API for Meter Data provides the ability for nMarket to receive meter reads from a participant’s external systems. The participant is responsible for sending the data according to the requirements specified in this API.

nMarket will store the data in market defined tables. One version can be stored for each source per date and interval.  Meter Read Sources can be defined in Codes.

nMarket will store data at either a Meter level or Transaction Point level defined in the Import/Export Process record, with the defaults as:

* MISO– will load Meter Data at the Transaction Point level.

Meter Data records are stored in nMarket in the MC\_METER\_DATA tables.

***Versioning***

For markets that require it, Meter Data Information is versioned in nMarket through the use of the Data Source Type field. Data loaded for one data source type will not affect existing records for other data source types. However, if meter data to be loaded into nMarket matches existing meter reads by participant, market, meter read type, location name, data source, date and time, the existing meter data in nMarket will be replaced with the new meter reads.

***General Conditions***

All energy units must be in MWh.  Each market will support hourly intervals (including a range of hours) for Meter Reads.

1. The ACTION field with values of INSERT, UPDATE, or DELETE has to be provided to determine

what data manipulation should occur for that Parameter in nMarket. The DML is based on the unique key for the Meter Data. The unique key includes the following fields:

* Participant Code
* Market Code
* Start Date
* Start Time
* End Date
* End Time
* Meter Read Source
* Meter Read Type
* Location Name

2. If new data is entered into the external system, nMarket will receive an action of INSERT or UPDATE and the new data will be inserted into nMarket with a status of NEW.

* If the data does not exist the record will be inserted.
* If data exists, the existing meter read records will be deleted and the new meter read records will be inserted.

3. If meter data is updated in the external system, nMarket should receive an action of UPDATE. To update any non-key fields, such as quantity, an update needs to be sent with the updated value and all other key values equal to the values in the original record.

Status = NEW; Status remains NEW

Status = UPD; Status remains UPD

Status = ERR, UNK or SUB; Status changes to UPD

4. To update any key fields, first a delete needs to be sent for that record, and then an insert needs to be sent for the new record with desired changes.

5. If a record is deleted in the external system, nMarket will receive an action of DELETE.

* Any existing record in nMarket where "Delete submitted records" configuration parameter is set to "False", then the record cannot be deleted else:  error message
* If the record exists in nMarket, the record will be physically deleted from the nMarket Database.
* If the record sent by the external system is not found in nMarket, the following error message will be generated and no action will be performed for that record.

“Record cannot be deleted because the record does not exist for <Location Name>for the period <Operating Day and Hour>.”

* Any existing record in nMarket will remain active unless a Delete is specifically issued by the external system.

6. Multiple actions should not be included in the same file for records having the same key values. For instance, it is not permissible to delete and then re-insert records for the same key values in the same API file. If such an operation is required, it should be performed in two separate API files.

7. Any tags included in the API file not explicitly defined by this document should be ignored.

***Rollback/Commits***

The load process will have two options when loading a file and encountering an error. In the

X\_IMPORTEXPORT\_CONFIG table, if RAISE\_ERROR\_NUM is set to 0, records that failed to load will not abort the load. If RAISE\_ERROR\_NUM is set to 1, records that failed to load will result in the load being rolled back.

***Process Code***

The process code for the Meter Data API is MCMTRRD. This code will be used as the process name to identify the process in the File Listener, HTTP Listener, and Web Services.

***Trigger Events***

The Meter Data interface has four ways in which it can be initiated to load the XML file.

* A manual load can be executed by any authorized user logged into nMarket via the load from file process.
* An automated load can be executed by the source system by writing a file to the file listener directory. nMarket will be equipped with a listener to look for the completed XML file and automatically load it into nMarket.
* A programmatic load can be executed by submitting a file to the Application Server via http (HTTP Listener).
* A programmatic load can be executed by submitting a file to the Application Server via Web Services.

***Meter Data API Field Descriptions/Data Mapping***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **API Data Element** | **XML Tag** | **Required** | **Valid Values** | **Format** | **nMarket Database Mapping** | **Notes** |
| Market Code | MKT\_CD | Yes | Midwest, MW, MISO | String |  | The market code associated w/ the Meter Read. The value must be set to the appropriate *market* to load Meter Read Data into nMarket. |
| Action | ACTION | Yes | INSERT, UPDATE, DELETE | String (6) |  | This is a required field that tells nMarket if this record is a new record to be inserted, an update to a previously inserted record or a request to delete a previously inserted or updated record. |
| Participant Code | PTCPT\_CD | Yes |  | String | MC\_METER\_DATA.PTCPT\_ID | This will be the configured code of the Participant. |
| Location Name | METER\_ID | Yes | Market Operator Defined Transaction Point Name or ISO ID |  | MC\_METER\_DATA.XP\_ID | **MISO:** The Market Operator transaction point is any Market Operator registered point (generator, zone, bus etc.) where a transaction/settlement can occur. This is the ISO defined ID for the Transaction Point or the Transaction Point Name. |
| Meter Read Type | METER\_RD\_TYP | Yes | GEN, LOAD | String | MC\_METER\_DATA\_TYPE.METER\_DATA\_TYPE\_CD    MC\_METER\_DATA.METER\_DATA\_TYPE\_CD |  |
| Meter Source Type | DATA\_SRC\_TYP | Yes | ISO, Internal, or any value that has been defined as a valid Meter Source Codes | String | MC\_METER\_DATA\_SOURCE\_TYPE.METER\_DATA\_SOURCE\_CD    MC\_METER\_DATA.METER\_DATA\_SOURCE\_CD | Meter Source Type is a description of the source of the meter read data.  **Do not implement - NOTE MISO**: To support submitting and versioning meter data for settlement purposes, users should load meter data into nMarket with sources of ‘07-Internal’ or ‘INT7’, ‘14-Internal’ or ‘INT14’, ‘55-Internal’ or ‘INT55’, ‘105-Internal’ or ‘INT105’. In turn, these meter data should be submitted for the appropriate settlement period that matches with the meter sources. This will ensure that meter data used during settlement are the meter data that users had submitted for the correct timeframe and can be comparable to the ISO settlement statements issued for S7, S14, S55 and S105. |
| Start Date | START\_DT | Yes |  | String (YYYYMMDD) | MC\_METER\_DATA.  START\_DT | The start date of the meter read |
| Start Time | START\_TIME | Yes |  | String  (HH24MISS)   [D] | MC\_METER\_DATA.  START\_DT | The start time of the meter read (the start hour or beginning hour). |
| End Date | END\_DT | Yes |  | String  (YYYYMMDD) | MC\_METER\_DATA.TIME\_INTRVL\_TYPE\_CD = 60    MC\_METER\_DATA.DT\_INTRVL\_VAL = +0000000 01:00:00.0 | The end date of the meter read  NOTE:  if range of hours, make it hourly intervals in database |
| End Time | END\_TIME | Yes |  | String  (HH24MISS)   [D] | MC\_METER\_DATA.TIME\_INTRVL\_TYPE\_CD = 60    MC\_METER\_DATA.DT\_INTRVL\_VAL = +0000000 01:00:00.0 | The end time of the meter read (the end hour or ending hour).  NOTE:  if range of hours, make it hourly intervals in database |
| Time Zone | TIME\_ZONE | Yes | GMT, EST, CST, MST, and PST | String (3) |  | The following time zones will be supported GMT, EST, CST, MST, and PST. The time zone tag will be used to specify the time zone for all date/time information in the interface. For example, if the start time is in EST, the end time must also be EST. |
| Meter Quantity | QUANTITY | Yes |  | Number (10v5) | MC\_METER\_DATA.QUANTITY | The actual meter read value from the meter in MWh.  **MISO**: Generation meter data should be loaded as negative values to be submitted to the ISO. |
| External ID | EXT\_ID | No |  | String | MC\_METER\_DATA.EXT\_ID | Optional field that is used to store a source system’s id that describes this record. |
| Meter Status | STATUS | No | A or "" = Actual  E = Estimated  M = Missing  P = Profile | String (1) | MC\_METER\_READ\_STATUS\_TYPE.METER\_READ\_STATUS\_CD    MC\_METER\_DATA.METER\_READ\_STATUS\_CD |  |
| Row ID | ROW\_ID | No |  | String |  | Optional field that is used to identify this record in error messages. |
|  |  |  |  |  | MC\_METER\_DATA.STATUS\_CD | Status should be "NEW" if the data has not ever been submitted to the ISO, if the previous data that is being overwritten has a status of "ERR", "UNK", "UPD" or "SUB" then the updated record status should change to "UPD" |
|  |  |  |  |  | MC\_METER\_DATA.START\_DAY\_DT | Trade Date in file    NOTE:  Trigger to help with index for performance. |

***Error Handling***

All errors generated by the API will be identified with the following parameters from the erroneous record (if populated): row\_id=<row\_id>, ext\_id=<ext\_id>, mkt\_cd=<mkt\_cd>, ptcpt\_cd = <ptcpt\_cd>, location\_name =<location\_name>, meter\_read\_typ=<meter\_read\_typ>, meter\_read\_source = <meter\_read\_source>, operating\_hour=<operating\_hour>

If the following conditions exist, the record for that interval should not be loaded and the following error messages should be generated:

* Another load of the same type is occurring.
  + The Meter Data file <filename> cannot be loaded because another Meter Data file is being loaded.
* Action is invalid.
  + Invalid Data passed from the file. Action <action> is invalid.
* Action is missing.
  + Insufficient number of parameters is being passed. Action is a required field.
* Delete action is invalid.
  + Invalid Data passed from the file. The action code passed is DELETE but no matching data exists.
* Market Code is missing
  + Insufficient number of parameters is being passed. Market\_cd is a required field.
* Invalid Market Code value
  + Invalid Data passed from file. Market Code <mkt\_cd> is invalid.
* Participant Code is missing.
  + Insufficient number of parameters is being passed. Ptcpt\_cd is a required field.
* Invalid Participant Code value.
  + Invalid Data passed from file. Participant <ptcpt\_cd> is invalid.
* Location Name is missing.
  + Insufficient number of parameters is being passed. Location Name is a required field.
* Invalid Location Name
  + Invalid Data passed from the file. Location Name <location\_name> is invalid.
* Meter Read Type is missing
  + Insufficient number of parameters is being passed. Meter Read Type is a required field.
* Invalid Meter Read Type
  + Invalid Data passed from the file. Meter Read Type <meter\_ready\_type> is invalid.
* Meter Read Source is missing
  + Insufficient number of parameters is being passed. Meter Read Source is a required field.
* Invalid Meter Read Source
  + Invalid Data passed from the file. Data Source Type <meter\_read\_source> is invalid.
* Start Date is missing
  + Insufficient number of parameters is being passed. Start Date is a required field.
* Start Date is invalid
  + Invalid Data passed from the file. Start Date is invalid.
* Start Time is missing
  + Insufficient number of parameters is being passed. Start Time is a required field.
* Start Time is invalid
  + Invalid Data passed from the file. Start Time is invalid.
* End Date is missing
  + Insufficient number of parameters is being passed. End Date is required.
* End Date is invalid
  + Invalid Data passed from the file. End Date is invalid.
* End Time is missing
  + Insufficient number of parameters is being passed. End Time is required.
* End Time is invalid
  + Invalid Data passed from the file. End Time is invalid.
* Invalid Date Combination (i.e. End Date starts before Start Date)
  + Invalid Data passed from the file. Start Date <start\_dt> and Start Time <start\_tm> is more than/equal to the End Date <end\_dt> and End Time <end\_tm>.
* Invalid Time Combination (i.e. End Time starts before Start Time)
  + Invalid Data passed from the file. Start Date <start\_dt> and Start Time <start\_tm> is more than/equal to the End Date <end\_dt> and End Time <end\_tm>.
* Time Zone is missing
  + Insufficient number of parameters is being passed. Time Zone is required.
* Time Zone is invalid
  + Invalid Data passed from the file. Time Zone is invalid.
* Quantity is missing
  + Insufficient number of parameters is being passed. Quantity is a required field.
* Invalid Quantity
  + Invalid data passed from file. Quantity <quantity> is an invalid value.
* Invalid Status
  + Invalid data passed from file. Status <status> is an invalid value.

**Data File Specification (DTD)**

<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT CROSS\_MARKET (METER\_DATA\*)>

<!ELEMENT METER\_DATA (PTCPT\_CD, MKT\_CD,  METER\_ID, METER\_RD\_TYP, DATA\_SRC\_TYP, READING+)>

<!ELEMENT READING (START\_DT, START\_TIME, END\_DT, END\_TIME, TIME\_ZONE, QUANTITY, STATUS?, ACTION, EXT\_ID?, ROW\_ID?)>

<!ELEMENT ACTION (#PCDATA)>

<!ELEMENT DATA\_SRC\_TYP (#PCDATA)>

<!ELEMENT END\_DT (#PCDATA)>

<!ELEMENT END\_TIME (#PCDATA)>

<!ELEMENT EXT\_ID (#PCDATA)>

<!ELEMENT METER\_ID (#PCDATA)>

<!ELEMENT METER\_RD\_TYP (#PCDATA)>

<!ELEMENT MKT\_CD (#PCDATA)>

<!ELEMENT PTCPT\_CD (#PCDATA)>

<!ELEMENT ROW\_ID (#PCDATA)>

<!ELEMENT QUANTITY (#PCDATA)>

<!ELEMENT START\_DT (#PCDATA)>

<!ELEMENT START\_TIME (#PCDATA)>

<!ELEMENT TIME\_ZONE (#PCDATA)>

<!ELEMENT STATUS (#PCDATA)>

**XML Schema Definition (XSD)**

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!--W3C Schema generated by XMLSpy v2008 sp1 (.altova.com)-->

<!--Please add namespace attributes, a targetNamespace attribute and import elements according to your requirements-->

<xs:schema xmlns:xs="<http://www.w3.org/2001/XMLSchema>" elementFormDefault="qualified">

<xs:import namespace="<http://www.w3.org/XML/1998/namespace>"/>

<xs:complexType name="CROSS\_MARKET">

<xs:sequence>

<xs:element ref="METER\_DATA" minOccurs="0" maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

<xs:element name="CROSS\_MARKET" type="CROSS\_MARKET"/>

<xs:complexType name="METER\_DATA">

<xs:sequence>

<xs:element ref="PTCPT\_CD"/>

<xs:element ref="MKT\_CD"/>

<xs:element ref="METER\_ID"/>

<xs:element ref="METER\_RD\_TYP"/>

<xs:element ref="DATA\_SRC\_TYP"/>

<xs:element ref="READING" maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

<xs:element name="METER\_DATA" type="METER\_DATA"/>

<xs:complexType name="READING">

<xs:sequence>

<xs:element ref="START\_DT"/>

<xs:element ref="START\_TIME"/>

<xs:element ref="END\_DT"/>

<xs:element ref="END\_TIME"/>

<xs:element ref="TIME\_ZONE"/>

<xs:element ref="QUANTITY"/>

<xs:element ref="STATUS" minOccurs="0"/>

<xs:element ref="ACTION"/>

<xs:element ref="EXT\_ID" minOccurs="0"/>

<xs:element ref="ROW\_ID" minOccurs="0"/>

</xs:sequence>

</xs:complexType>

<xs:element name="READING" type="READING"/>

<xs:complexType name="ACTION" mixed="true"/>

<xs:element name="ACTION" type="ACTION"/>

<xs:complexType name="DATA\_SRC\_TYP" mixed="true"/>

<xs:element name="DATA\_SRC\_TYP" type="DATA\_SRC\_TYP"/>

<xs:complexType name="END\_DT" mixed="true"/>

<xs:element name="END\_DT" type="END\_DT"/>

<xs:complexType name="END\_TIME" mixed="true"/>

<xs:element name="END\_TIME" type="END\_TIME"/>

<xs:complexType name="EXT\_ID" mixed="true"/>

<xs:element name="EXT\_ID" type="EXT\_ID"/>

<xs:complexType name="METER\_ID" mixed="true"/>

<xs:element name="METER\_ID" type="METER\_ID"/>

<xs:complexType name="METER\_RD\_TYP" mixed="true"/>

<xs:element name="METER\_RD\_TYP" type="METER\_RD\_TYP"/>

<xs:complexType name="MKT\_CD" mixed="true"/>

<xs:element name="MKT\_CD" type="MKT\_CD"/>

<xs:complexType name="PTCPT\_CD" mixed="true"/>

<xs:element name="PTCPT\_CD" type="PTCPT\_CD"/>

<xs:complexType name="ROW\_ID" mixed="true"/>

<xs:element name="ROW\_ID" type="ROW\_ID"/>

<xs:complexType name="QUANTITY" mixed="true"/>

<xs:element name="QUANTITY" type="QUANTITY"/>

<xs:complexType name="START\_DT" mixed="true"/>

<xs:element name="START\_DT" type="START\_DT"/>

<xs:complexType name="START\_TIME" mixed="true"/>

<xs:element name="START\_TIME" type="START\_TIME"/>

<xs:complexType name="TIME\_ZONE" mixed="true"/>

<xs:element name="TIME\_ZONE" type="TIME\_ZONE"/>

<xs:complexType name="STATUS" mixed="true"/>

<xs:element name="STATUS" type="STATUS"/>

</xs:schema>

**MISO**

<CROSS\_MARKET>

<METER\_DATA>

<PTCPT\_CD>TEST</PTCPT\_CD>

<MKT\_CD>MISO</MKT\_CD>

<METER\_ID>UN.LINCOLN13.8LINCOLN\_1</METER\_ID>

<METER\_RD\_TYP>GEN</METER\_RD\_TYP>

<DATA\_SRC\_TYP>Internal</DATA\_SRC\_TYP>

<READING>

<START\_DT>20030501</START\_DT>

<START\_TIME>000000</START\_TIME>

<END\_DT>20030501</END\_DT>

<END\_TIME>010000</END\_TIME>

<TIME\_ZONE>EST</TIME\_ZONE>

<QUANTITY>250.00</QUANTITY>

   <STATUS> </STATUS>

   <ACTION>INSERT</ACTION>

   <EXT\_ID>500000</EXT\_ID>

<ROW\_ID>1</ROW\_ID>

</READING>

</METER\_DATA>

<METER\_DATA>

<PTCPT\_CD>TEST</PTCPT\_CD>

<MKT\_CD>MIDWEST</MKT\_CD>

<METER\_ID>UN.LINCOLN13.8LINCOLN\_1</METER\_ID>

<METER\_RD\_TYP>GEN</METER\_RD\_TYP>

<DATA\_SRC\_TYP>ISO</DATA\_SRC\_TYP>

<READING>

<START\_DT>20030501</START\_DT>

<START\_TIME>000000</START\_TIME>

<END\_DT>20030501</END\_DT>

<END\_TIME>010000</END\_TIME>

<TIME\_ZONE>EST</TIME\_ZONE>

<QUANTITY>150.00</QUANTITY>

   <STATUS> </STATUS>

   <ACTION>INSERT</ACTION>

   <EXT\_ID>500000</EXT\_ID>

<ROW\_ID>1</ROW\_ID>

</READING>

</METER\_DATA>

</CROSS\_MARKET>

***nMarket Data***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **METER\_DATA\_ID** | **METER\_DATA\_TYPE\_CD** | **PTCPT\_ID** | **METER\_DATA\_SOURCE\_CD** | **TIME\_INTRVL\_TYPE\_CD** | **XP\_ID** | **START\_DT** | **DT\_INTRVL\_VAL** | **QUANTITY** | **EXT\_ID** | **METER\_ID** | **STATUS\_CD** | **METER\_READ\_STATUS\_CD** | **INACTIVE\_FLG** | **USER\_ID** | **REVISED\_DT** | **ROW\_ID** | **START\_DAY\_DT** |
| 200 | GEN | 15 | INTRNL | 60 | 1064 | 2003May01 05:00:00 | +0000000 01:00:00.0 | 250 | 500000 | 10641 | NEW | ACTUAL | 0 | -10 | 2015Jun22 09:22:35 |  | 2003May01 05:00:00 |
| 201 | GEN | 15 | ISO | 60 | 1064 | 2003May01 05:00:00 | +0000000 01:00:00.0 | 150 | 500000 | 10641 | NEW | ACTUAL | 0 | -10 | 2015Jun22 09:22:35 |  | 2003May01 05:00:00 |

***Outstanding Questions***

* Do I store the Meter Id associated to the XP in the Meter\_ID column or do I remove column  -- the old product does not store meters in the meter\_id column - when creating a Meter in the UI you have the ability to choose a Meter which means we need to identify the Meter ID associated to the Transaction Point and insert it into the Meter\_ID column.
* If I need the Meter ID - why not allow the user to use the Meter\_ID in the API - still need to determine
* Where does the tag info go in message log
* How is internal data submitted?  Is it just for Meter Source Cd of Internal or can a user submit other Meter Source Cd (excluding ISO)?  -- looks like INT7 and INT14 can be submitted?
* Can we have multiple participants in the file? Yes, you can multiple participants in a file but we don't recommend it to our clients.  IF the client puts multiple participants then more file will go in the 1st participants listed auditpath.  This scenario does not need to be tested since it is an unlikely scenario.

***Testing Approach***

|  |  |
| --- | --- |
| * + **API LOADS** | **(A)utomated / (M)anual** |
| Load from File | M |
| FileListener | On hold |
| Inclusive of all optional & required fields | A |
| INSERT, UPDATE, DELETE (all day & partial) | A |
| Only Required fields | A |
| Test all Time Zone combinations | A |
| Unique Key Validation (verifying duplications don’t exist) (ex: per Offer type / Sub Type) | M |
| Validate end-dated XP validation | M |
| Validate ISO & Internal API duplicate data with different Sources can exist w/o wiping each other out | M |
| Validate Start Date is always less than the End Date | M |
| Validate each valid Participant type for all valid XP types | M |
| Error handling- Duplicate data validation for data integrity | M |
| Error handling - Formatting | M |
| Verify a NEW record when updated stays with a NEW status | M |
| Verify an ERR status record when updated changes to UPD status | M |
| Verify an UNK status record when updated changes to UPD status | M |
| Verify an SUB status record when updated changes to UPD status | M |
| Verify an UPD status record when updated remain UPD status | M |
| Verify when loading a new record status is NEW status | M |
| Verify only SUB records can be deleted when the "Delete submitted records" configuration parameter is set to "True" | M |
| Verify SUB status records cannot be deleted when the "Delete submitted records" configuration parameter is set to "False" | M |