IEC 62541-11

Edition 3.0 2025-12

INTERNATIONAL STANDARD

OPC unified architecture -
Part 11: Historical Access

ICS 25.040.40; 35.100.05

ISBN 978-2-8327-0846-0

IEC 62541-11:2025-12(en)

IEC Secretariat
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee, …). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

IEC Products & Services Portal - products.iec.ch
Discover our powerful search engine and read freely all the publications previews, graphical symbols and the glossary. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 500 terminological entries in English and French, with equivalent terms in 25 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

Warning! Make sure that you obtained this publication from an authorized distributor.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

OPC unified architecture - Part 11: Historical Access

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) IEC draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). IEC takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, IEC had not received notice of (a) patent(s), which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at https://patents.iec.ch. IEC shall not be held responsible for identifying any or all such patent rights.

IEC 62541-11 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation. It is an International Standard.

This fourth edition cancels and replaces the third edition published in 2020. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

a) a functionality has been added to support retrieving of modified events;
b) an Event has been added to indicate when a backfill occurred;
c) a new ReferenceType that can be used to indicate an external node has been defined;
d) the text has been improved to better explain the concept of annotation and remove conflicting explanations;

e) a default historian configuration (and where to find it) has been defined;

f) HistoricalEventConfigurationType, which provides general configuration information about the historical Event storage, has been added;

g) the text has been updated and optional fields have been added to HA configuration object to allow configuration to be defined for periodic data collection, not just for exception-based collection;

h) an ObjectType that can be used for external event collection has been provided as well as an example how historians can be configured.

The text of this International Standard is based on the following documents:

Draft	Report on voting
65E/1058/CDV	65E/1096/RVC

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/publications.

Throughout this document and the other parts of the IEC 62541 series, certain document conventions are used:

Italics are used to denote a defined term or definition that appears in the "Terms and definitions" clause in one of the parts of the IEC 62541 series.

Italics are also used to denote the name of a service input or output parameter or the name of a structure or element of a structure that are usually defined in tables.

The *italicized terms and names* are, with a few exceptions, written in camel-case (the practice of writing compound words or phrases in which the elements are joined without spaces, with each element's initial letter capitalized within the compound). For example, the defined term is *AddressSpace* instead of Address Space. This makes it easier to understand that there is a single definition for *AddressSpace*, not separate definitions for Address and Space.

A list of all parts in the IEC 62541 series, published under the general title *OPC Unified Architecture*, can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

– reconfirmed,

– withdrawn, or

– revised.

1 Scope

This part of IEC 62541 belongs to the OPC Unified Architecture standards series and defines the *Information Model* associated with Historical Access (HA). It particularly includes additional and complementary descriptions of the *NodeClass*es and *Attributes* needed for Historical Access, additional standard *Properties*, and other information and behaviour.

The complete *AddressSpace* Model including all *NodeClass*es and *Attributes* is specified in IEC 62541-3. The predefined *Information Model* is defined in IEC 62541-5. The *Services* to detect and access historical data and events, and description of the *ExtensibleParameter* types are specified in IEC 62541-4.

This document includes functionality to compute and return *Aggregates* like minimum, maximum, average etc. The *Information Model* and the concrete working of *Aggregates* are defined in IEC 62541-13.

Conventions for Historical Access Clients are informatively provided in Annex A.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 62541-1, *OPC Unified Architecture - Part 1: Overview and Concepts*

IEC 62541-3, *OPC Unified Architecture - Part 3: Address Space Model*

IEC 62541-4, *OPC Unified Architecture - Part 4: Services*

IEC 62541-5, *OPC Unified Architecture - Part 5: Information Model*

IEC 62541-7, *OPC Unified Architecture - Part 7: Profiles*

IEC 62541-8, *OPC Unified Architecture - Part 8: Data Access*

IEC 62541-13, *OPC Unified Architecture - Part 13: Aggregates*

3 Terms, definitions and abbreviations

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62541-1, IEC 62541-3, IEC 62541-4, IEC 62541-13 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

– IEC Electropedia: available at https://www.electropedia.org/
– ISO Online browsing platform: available at https://www.iso.org/obp

3.1.1
Annotation
metadata associated with an item at a given instance in time

Note 1 to entry: An *Annotation* is metadata that is associated with an item at a given instance in time.

3.1.2
BoundingValues
values associated with the starting and ending time

Note 1 to entry: *BoundingValues* are the values that are associated with the starting and ending time of a *ProcessingInterval* specified when reading from the historian. *BoundingValues* can be required by *Clients* to determine the starting and ending values when requesting *RawData* over a time range. If a *RawData* value exists at the start or end point, it is considered the bounding value even though it is part of the data request. If no *RawData* value exists at the start or end point, then the *Server* will determine the boundary value, which possibly requires data from a data point outside of the requested range. See 4.6 for details on using *BoundingValues*.

3.1.3
Historian
application storing time series data and/or time series events

3.1.4
HistoricalNode
Object, *Variable*, *Property* or *View* in the *AddressSpace* where a *Client* can access historical data or *Events*

Note 1 to entry: A *HistoricalNode* is a term used in this document to represent any *Object*, *Variable, Property* or *View* in the *AddressSpace* for which a *Client* can read and/or update historical data or *Events*. The terms "*HistoricalNodes's* history" or "history of a *HistoricalNodes*" will refer to the time series data or *Events* stored for this *HistoricalNode*. The term *HistoricalNode* refers to both *HistoricalDataNodes* and HistoricalEventNodes.

3.1.5
HistoricalDataNode
Variable or *Property* in the *AddressSpace* where a *Client* can access historical data

Note 1 to entry: A *HistoricalDataNode* represents any Variable or Property in the *AddressSpace* for which a *Client* can read and/or update historical data. "*HistoricalDataNode* history" or "history of a *HistoricalDataNode*" refers to the time series data stored for this *HistoricalNode*. Examples of such data are:

- device data (like temperature sensors)
- calculated data
- status information (open/closed, moving)
- dynamically changing system data (like stock quotes)
- diagnostic data

The term *HistoricalDataNodes* is used when referencing aspects of the standard that apply to accessing historical data only.

3.1.6
HistoricalEventNode
Object or *View* in the *AddressSpace* for which a *Client* can access historical *Events*

Note 1 to entry: "*HistoricalEventNode's* history" or "history of a *HistoricalEventNode*" refers to the time series *Events* stored in some historical system. Examples of such data are:

- *Notifications*
- system *Alarms*
- operator action *Events*
- system triggers (such as new orders to be processed)

The term *HistoricalEventNode* is used when referencing aspects of the standard that apply to accessing historical *Events* only.

3.1.7
ModifiedValues
HistoricalDataNode's value that has been changed (or manually inserted or deleted) after it was stored in the historian

Note 1 to entry: For some *Servers*, a lab data entry value is not a *modified value*, but if a user corrects a lab value, the original value would be considered a *modified value*, and would be returned during a request for *ModifiedValues*. Also manually inserting a value that was missed by a standard collection system can be considered a *modified value*. Unless specified otherwise, all historical *Services* operate on the current, or most recent, value for the specified *HistoricalDataNode* at the specified timestamp. Requests for *ModifiedValues* are used to access values that have been superseded, deleted or inserted. It is up to a system to determine what is considered a *modified value*. Whenever a *Server* has modified data available for an entry in the historical collection it shall set the *ExtraData* bit in the *StatusCode.*

3.1.8
RawData
data that is stored within the historian for a *HistoricalDataNode*

Note 1 to entry: The data can be all data collected for the *DataValue* or it can be some subset of the data depending on the historian and the storage rules invoked when the item's values were saved.

3.1.9
StartTime/EndTime
bounds of a history request which define the time domain

Note 1 to entry: For all requests, a value falling at the end time of the time domain is not included in the domain, so that requests made for successive, contiguous time domains will include every value in the historical collection exactly once.

3.1.10
TimeDomain
interval of time covered by a particular request, or response

Note 1 to entry: In general, if the start time is earlier than or the same as the end time, the time domain is considered to begin at the start time and end just before the end time; if the end time is earlier than the start time, the time domain still begins at the start time and ends just before the end time, with time "running backward" for the particular request and response. In both cases, any value which falls exactly at the end time of the *TimeDomain* is not included in the *TimeDomain*. See the examples in 4.6. *BoundingValues* effect the time domain as described in 4.6.

All timestamps which can legally be represented in a *UtcTime DataType* are valid timestamps, and the *Server* has the choice to, or not to return an invalid argument result code due to the timestamp being outside of the range for which the *Server* has data. See IEC 62541-3 for a description of the range and granularity of this *DataType*. *Servers* are expected to handle out-of-bounds timestamps gracefully and return the proper *StatusCodes* to the *Client*.

3.1.11
StructuredHistoryData
structured data stored in a history collection where parts of the structure are used to uniquely identify the data within the data collection

Note 1 to entry: Most historical data applications assume only one current value per timestamp. Therefore, the timestamp of the data is considered the unique identifier for that value. Some data or meta data such as *Annotations* can permit multiple values to exist at a single timestamp. In such cases the *Server* would use one or more parameters of the *StructuredHistoryData* entry to uniquely identify each element within the history collection. *Annotations* are examples of *StructuredHistoryData*.

3.2 Abbreviated terms

DA Data Access

HA Historical Access

HDA Historical Data Access

UA Unified Architecture

4 Concepts

4.1 General

This document defines the handling of historical time series data and historical *Event* data in the OPC Unified Architecture (in a Historian). Included is the specification of the representation of historical data and *Events* in the *AddressSpace*.

4.2 Data architecture

A *Server* supporting Historical Access provides *Clients* with transparent access to different historical data and/or historical *Event* sources (e.g., process *Historians*, event *Historians*, etc.).

The historical data or *Events* can be in a proprietary data collection, database or a short-term buffer within memory. A *Server* supporting Historical Access will provide historical data and *Events* for all or a subset of the available *Variables*, *Objects, Properties* or *Views* within the *Server AddressSpace*.

Figure 1 illustrates how the *AddressSpace* of a UA *Server* can consist of a broad range of different historical data and/or historical *Event* sources.

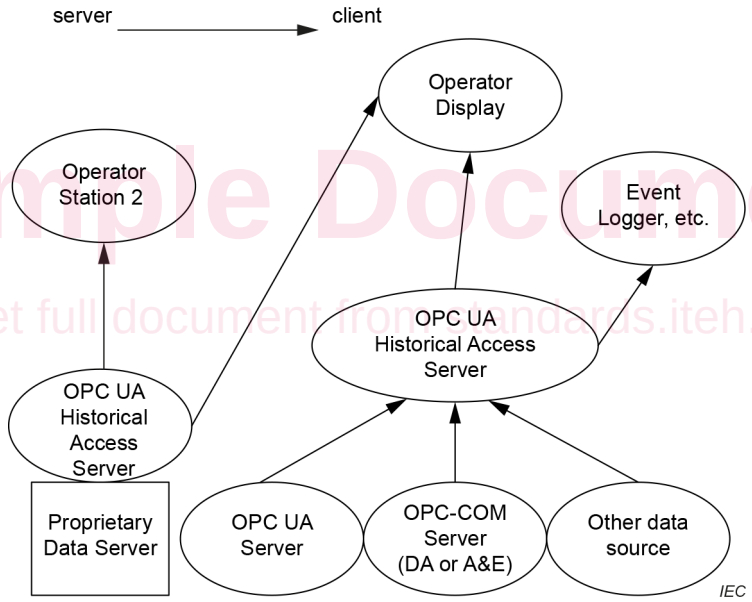

Figure 1 – Possible OPC UA Server supporting Historical Access

The *Historian* can be implemented as a standalone OPC UA *Server* that collects data from another OPC UA *Server* or another data source. The *Historian* can also just aggregate historical data from underlying *Historians*. The *Client* that references the OPC UA *Server* supporting Historical Access for historical data can be simple trending packages that desire values over a given time frame or they can be complex reports that require data in multiple formats.

There are general requirements for *Historians*, but *Historians* can vary in functionality. A consistent requirement for all *Historians* is that they store Historical data including a timestamp. All historical data should include status information for each value, but a *Historian* can compress this to only storing status information that indicates a problem (Bad status) and/or status change, instead of storing a status for every time series data item. The status of historical data can be complex. What is required is that the values returned as part of the timeseries raw data match the data that would have been observed if the *Value* was subscribed to at that point in time.

Historical *Events* are more complicated. In a stream of *Events* each Event can have a different list of fields. *EventTypes* are defined in a hierarchical manner, where each *EventType* inherits fields from its parent type and can add additional fields. Some of these additional fields can be mandatory and are required to understand or process the given *EventType*. A *Historian* that stores *Events*, shall be configurable to store all mandatory fields for any *EventTypes* that it historizes. If it receives for storage an *EventType* that it does not support all mandatory fields for, it can store it as one of its supertype *EventTypes* (one that it does support all mandatory fields for), but then it shall not claim that it supports historizing of that *EventType* (see 5.4.3). The *Historian* shall also provide information about the fields that are currently being historized (see 5.4.3).

4.3 Historians and interruption of data collection

When an *Historian* is collecting and storing data, the data collection can be interrupted. The interruption can have been for collecting the current values of data or for an event stream. The interruption can have been due to an interruption in the source of a value or an interruption of the forwarding of historical data from an underlying *Historian*. The interruption can also have been due to an action that stopped the collection of *HistoricalData* or historical *Events*. Some of these interruptions can recover with no loss of data, others can result in data gaps. The *Historian* shall report any gaps when a client is accessing the stored historical data with an error code of *Bad_DataLost*.

For example, if a subscription for data breaks, and the historian recovers the subscription after several minutes, it can check the *SourceTimestamp* of the initial values in the subscription. If the initial value *SourceTimestamp* matches the last stored value *SourceTimestamp*, then no data was lost and nothing needs to be stored indicating the given *HistoricalDataNode*'s data collection was interrupted. But if the *SourceTimestamp* of the initial value is later than the *SourceTimestamp* of the last stored value, then the historian has no way of knowing if any data was lost and it shall record a bad status for the value with the timestamp of when the connection was broken.

For *Event* collection, in addition to what is described for a data collection the following also has to be addressed. If a *Subscription* for *Event* is interrupted for long enough that the *Subscription* buffer reports an overflow (instance of *EventQueueOverflowEventType*), then the *Event* storage shall report this error. This can be accomplished by storing the instance of *EventQueueOverflowEventType* event or by recording a *Bad_DataLost.* If the lost data is indicated by *EventQueueOverflowEventType*, this event shall always be returned for any filter, just as it is in an *Event Subscription*.

4.4 Modification of Historical Data/Events

A *Historian* collects values (data or *Events*) and provides long term storage of these values. Occasionally data that is collected is incorrect. It can be incorrect for many reasons, such as a failed sensor, or failed communication or even an out of calibration sensor. Sometimes the correct values are obtained from another source and the historical value is updated from this source. In many *Historians*, the act of updating a historical value needs to be tracked. This specification provides two aspects for tracking:

- Recording the original value and information related to who made the change (see 6.9 for *HistoryUpdateDetails*).
- Generating an audit event describing the history edit action (see 5.8).

Some *Historians* obtain data from other sources (for example not an OPC UA *Subscription*). This data can be from lab data analysis package or other off-line activities. This data is typically provided in bulk and with *SourceTimestamps* that are in the past. For this type of data, even though it can be inserted using an *HistoryUpdateDetails* service call, the *Historian* does not have to create a modification record for the initial insertion.

Modification of *Events* is different than modification of data, in that in a single event stream there can be many *Events* with the same *Timestamp*. *Events* are identified by their EventId. The *EventId* is generated as part of normal *Event* processing in a *Server* and is used to correlate actions related to the *Event*. For example, when an Alarm (which is represented by an *Event)* is acknowledged, the acknowledgement is related to the specific *EventId* that represent that event is time. A *Historian* that historizes *Events* shall store the *EventId* that it receives; Modification of *Events* shall not allow modification to *EventIds.*

4.5 Timestamps

The nature of OPC UA Historical Access requires that a single timestamp reference be used to relate the multiple data points, and the *Client* can request which timestamp will be used as the reference. See IEC 62541-4 for details on the *TimestampsToReturn* enumeration. An OPC UA *Server* supporting Historical Access will treat the various timestamp settings as described below. A HistoryRead with invalid settings will be rejected with Bad_TimestampsToReturnInvalid (see IEC 62541-4).

For *HistoricalDataNodes*, the *SourceTimestamp* is used to determine which historical data values are to be returned.

The request is in terms of *SourceTimestamp* but the reply could be in *SourceTimestamp*, *ServerTimestamp* or both timestamps. If the reply has the *Server* timestamp the timestamps could fall outside of the range of the requested time.

SOURCE	Return the SourceTimestamp.
SERVER	Return the ServerTimestamp.
BOTH	Return both the SourceTimestamp and ServerTimestamp.
NEITHER	This is not a valid setting for any HistoryRead accessing HistoricalDataNodes.

Any reference to timestamps in this context throughout this document will represent either *ServerTimestamp* or *SourceTimestamp* as dictated by the type requested in the HistoryRead *Service*. Some *Servers* may not support historizing both *SourceTimestamp* and *ServerTimestamp*, but it is expected that all *Servers* will support historizing *SourceTimestamp* (see IEC 62541-7 for details on *Server Profiles*).

If a request is made requesting both *ServerTimestamp* and *SourceTimestamp* and the *Server* is only collecting the *SourceTimestamp,* the *Server* shall return Bad_TimestampNotSupported. Some *Historians* can aggregate data from underlying *Historians.* These *Servers* can have a mix of data, some with *ServerTimestamp* some without. The *Historian* may not know if a *HistoricalDataNode* supports *ServerTimestamps*. As a result of this uncertainty, the Bad_TimestampNotSupported can be returned on the service level or at the operation level.

For HistoricalEventNodes this parameter does not apply. This parameter is ignored since the entries returned are dictated by the *Event* Filter. See IEC 62541-4 for details.

4.6 Bounding Values and time domain

When accessing *HistoricalDataNodes via* the *HistoryRead Service*, requests can set a flag, returnBounds, indicating that *BoundingValues* are requested. For a complete description of the *Extensible Parameter HistoryReadDetails* that include *StartTime*, *EndTime* and NumValuesPerNode, see 6.5. The concept of Bounding Values and how they affect the time domain that is requested as part of the *HistoryRead* request is further explained in 4.4. Subclause 4.4 also provides examples of *TimeDomains* to further illustrate the expected behaviour.

When making a request for historical data using the *HistoryRead Service*, the required parameters include at least two of these three parameters: startTime, endTime and *numValuesPerNode*. What is returned when *BoundingValues* are requested varies according to which of these parameters are provided. For a *Historian* that has values stored at 5:00, 5:02, 5:03, 5:05 and 5:06, the data returned when using the Read Raw functionality is given by Table 1. In the table, FIRST stands for a tuple with a value of null, a timestamp of the specified *StartTime*, and a *StatusCode* of Bad_BoundNotFound. LAST stands for a tuple with a value of null, a timestamp of the specified *EndTime*, and a StatusCode of Bad_*BoundNotFound*.

In some cases, attempting to locate bounds, particularly FIRST or LAST points, can be resource intensive for *Servers*. Therefore, how far back or forward to look in history for Bounding Values is *Server* dependent, and the *Server* search limits can be reached before a bounding value can be found. There are also cases, such as reading *Annotations* or *Attribute* data where Bounding Values may not be appropriate. For such use cases it is permissible for the *Server* to return a *StatusCode* of *Bad_BoundNotSupported.*

Table 1 – Bounding Value examples

Start Time	End Time	numValuesPerNode	Bounds	Data Returned
5:00	5:05	0	Yes	5:00, 5:02, 5:03, 5:05
5:00	5:05	0	No	5:00, 5:02, 5:03
5:01	5:04	0	Yes	5:00, 5:02, 5:03, 5:05
5:01	5:04	0	No	5:02, 5:03
5:05	5:00	0	Yes	5:05, 5:03, 5:02, 5:00
5:05	5:00	0	No	5:05, 5:03, 5:02
5:04	5:01	0	Yes	5:05, 5:03, 5:02, 5:00
5:04	5:01	0	No	5:03, 5:02
4:59	5:05	0	Yes	FIRST, 5:00, 5:02, 5:03, 5:05
4:59	5:05	0	No	5:00, 5:02, 5:03
5:01	5:07	0	Yes	5:00, 5:02, 5:03, 5:05, 5:06, LAST
5:01	5:07	0	No	5:02, 5:03, 5:05, 5:06
5:00	5:05	3	Yes	5:00, 5:02, 5:03
5:00	5:05	3	No	5:00, 5:02, 5:03
5:01	5:04	3	Yes	5:00, 5:02, 5:03
5:01	5:04	3	No	5:02, 5:03
5:05	5:00	3	Yes	5:05, 5:03, 5:02
5:05	5:00	3	No	5:05, 5:03, 5:02
5:04	5:01	3	Yes	5:05, 5:03, 5:02
5:04	5:01	3	No	5:03, 5:02
4:59	5:05	3	Yes	FIRST, 5:00, 5:02
4:59	5:05	3	No	5:00, 5:02, 5:03
5:01	5:07	3	Yes	5:00, 5:02, 5:03
5:01	5:07	3	No	5:02, 5:03, 5:05
5:00	UNSPECIFIED	3	Yes	5:00, 5:02, 5:03
5:00	UNSPECIFIED	3	No	5:00, 5:02, 5:03
5:00	UNSPECIFIED	6	Yes	5:00, 5:02, 5:03, 5:05, 5:06, LAST[a]
5:00	UNSPECIFIED	6	No	5:00, 5:02, 5:03, 5:05, 5:06
5:07	UNSPECIFIED	6	Yes	5:06, LAST
5:07	UNSPECIFIED	6	No	NODATA