MCWS ReSTful API User's Guide

Guides

Prepared By:

Nicole M Witek — Nov 10, 2020
Nicole M Witek — Date
Author

Nicole M Witek — Nov 10, 2020
Nicole M Witek — Date
Custodian

Approved By:

Moojan Khazra — Nov 9, 2020
Moojan Khazra — Date
Task Manager (TM)

Document Release Authority:

Jennifer McCleary — Nov 10, 2020
Jennifer McCleary — Date
Document Release Authority

MGSS Document No: **DOC-001496 Rev L**
Issue Date: Nov 10, 2020
Effective Date: Nov 10, 2020

Configuration Item:Mission Control Web Service - 631.11

Jet Propulsion Laboratory
California Institute of Technology

Reviewers

Review solicited from the following and includes the required minimum reviewers (in bold) as specified in DOC-000016 Documentation Structure, Standards and Definitions. The Review Date identifies respondents but does not indicate approval or concurrence, only acknowledgement that the reviewer has reviewed the document, submitted recommendations/feedback, and are aware of the document and its impact. This page has been auto-generated by the Document Management System.

Review Date	Name / Title
	J. P. Pan Software Developer
Sep 24, 2020	**Eva Bokor** S/W Quality Assurance Engineer (SQAE)
Oct 1, 2020	**Moojan Khazra** AMMOS Element Manager (AEM) Multi-Mission Control Systems (MMCS)
	Bradley J Clement SSE Mission Control Subsystem (MC)
Oct 1, 2020	**Moojan Khazra** Task Manager (TM)

Document Change Log

Revision	Date Submitted	Affected Sections or Pages	Change Summary
Initial		All	Initial issue of the document.
A (was 2)	12 Feb 2016	section 2.3, section 3.3, section 4, section 5, section 6	• added ChannelLAD and ChannelMinMax Virtual Datatables • reworked stream interface • added Virtual Opaque Files • added columns to ChannelDictionary Virtual Datatable
B (was 3)	7 Mar 2017	Various	• updated websocket query filter parameter from session to session_id • minor clarifications and editorial corrections
C	28 Jul 2017	(1) Section 2.3 *MCWS Virtual Datatables and Opaque Files* (2) Section 4 *Accessing Realtime Content* (3) Section 5 *Accessing Virtual Opaque File Content* (4) Section 7 *Appendix: Virtual Datatable Defintions*	(1) Added information related to PacketSummaryEvent and CommandEvent Virtual Datatables. (2) Section 4.1 *Creating a Stream*: Added “PacketSummaryEvent” and “CommandEvent” to list of supported streams. (3) Added text related to DataProductContent Virtual Opaque File. (4a) Section 7.8 *Virtual Datatable Defintion – PacketSummaryEvent*: Added new Virtual Datatable specification. (4b) Section 7.*9 Virtual Datatable Defintion – Session*: Modified Virtual Datatable specification to include a new column labeled “venue_type”. (4c) Section 7.12 *Virtual Datatable Defintion – CommandEvent*: Added new Virtual Datatable specification.
D	14 Aug 2017	Section 7.2 *Virtual Datatable Definition - ChannelDictionary*	Added io_format and eu_io_format in support of Jira IDM-724.

E	30 Nov 2017	(1) Section 7 *Accessing Dictionary Content* (2) Section 8 *Appendix: Virtual Datatable Definitions* (3) Section 9: *Appendix: Examples*	(1) Section added in support of IDM-738 (Caching mechanism for dictionary queries) (2) Section number was 7 and was incremented to be 8. (3) Section number was 8 and was incremented to be 9.
F	1 May 2018	(1) Section 2.3 *MCWS Virtual Datatables and Opaque Files*	(1a) "tlm/ldb/ChannelEnumeration Dictionary/*dictionaryName*" added to *Telemetry Engineering / Housekeeping / Analysis (EHA) channel information* section (1b) "tlm/lom/CommandEvent" added to *Command Event information* section (1c) "tlm/lom/Frame" and "tlm/lom/Packet" added to *Other telemetry-related information* section (1d) "tlm/lom/Session" added to *AMPCS-specific information* section (1e) Virtual Datatables access patterns table updated to reflect changes 1a, 1b, 1c, and 1d.
		(2) Section 3.3 *LAD Queries*	(2) "See section 7…" updated to be "See section 8…" to reflect additional sections added since document was originally produced.
		(3) Section 3.3.1 *LAD Queries (Channel or EVR Virtual Datatable)*	(3) "…where ladTypeValue is one of "ert", "scet", or "rct"…" modified to be "…where ladTypeValue is one of "ert", "scet", or "event"…"
		(4) Section 8.3 *Virtual Datatable Definition — ChannelEnumerationDictionary*	(4) New section created in support of work Jira IDM-785 (ChannelEnumerationDictionary endpoint). Supports requirement IDM-811.
		(5) Section 8.5 *Virtual Datatable Definition – DataProduct*	(5) Entry added to Virtual Datatable: unique_name.
		(6) Section 8.3 *Virtual Datatable Defintion – ChannelEnumerationDictionary*	(6) Section moved such that it now follows *Virtual Datatable Defintion – ChannelDictionary*.

		(7) Section 8.4 *Virtual Datatable Defintion – CommandEvent* (8) Section 3.2 *Historical Queries* > 3.2.1 *Performing a Query*, and Section 4 *Accessing Realtime Content* > 4.2 *Reading from a Stream*	(7) Section moved such that it now follows *Virtual Datatable ChannelEnumerationDictionary* which now follows *Virtual Datatable Defintion – ChannelDictionary*. (8) Added information re: MCWS behavior re: escaped double-quote and backslash characters, i.e., that MCWS will escape instances of these characters. (Note: Updates were also made in the *MCWS RESTful SIS*, Rev C (DOC-001489) re: the behavior data within datatables.)
G	20 September 2018	(1) Section 2.3 *MCWS Virtual Datatables* (1) Section 3.3.2 *LAD Queries* (2) Section 8.13 *Virtual Datatable Definition – ChannelMinMax*	(1) Changed references from lad/ChannelLAD to lom/ChannelLAD (2) Added more column names and descriptions (session_id, session_host, dn, eu, dn_alarm_state, eu_alarm_state)
Rev G, Approval	10 October 2018	(1) Section 1.2 *Why is this User's Guide needed?*	(1) Replaced "AMPCS-managed" with "Mission Control" and added "telemetry and telecommand"
H	24 May 2019	(1) Section 2.3 *MCWS Virtual Datatables and Opaque Files* (2) Section 8.5 *Virtual Datatable Definition – DataProduct* (3) Section 8.9 *Virtual Datatable Definition – FrameSummaryEvent* (4) Section 8.11 *Virtual Datatable Definition –*	(1) Added FrameSummaryEvent (2) Updated description for record_type field (3) New section (4) Corrected record_type description from 'packet_summary' to 'packet_summary_event'

		PacketSummaryEvent	
Rev H, Approval	29 June 2019	(1) Section 1.3 *Controlling Documents and Processes* (2) Section 8.5 *Virtual Datatable Definition -- DataProduct*	(1) Updated SIS Rev from C to E and date it was released. (2) Changed description of ground_status field to list literal String values.
Rev I	7 October 2019	(1) Section 2.3 *MCWS Virtual Datatables and Opaque Files* (2) Section 4 *Accessing Realtime Content* (3) Section 5 *Accessing Virtual Opaque File Content* (4) Section 8.5 *Virtual Datatable Definition – DataProduct* (5) Section 8.10 *Virtual Datatable Definition – Message*	(1) Added stream/Message (2) Added “FrameSummaryEvent” and “Message” to list of supported streams. (3) Added a paragraph clarifying that product files must be accessible on the MCWS host. (4) Added “product_part”, “part_number”, “reason”, “log”, “creation_time” and “event_time” column and description. Put them at the end with a special heading to signal that these are specific to realtime. Added note that “transaction_id” column is only populated in realtime data. Added text to signal which fields are populated for “product_started” records. (5) New Section
Rev I, Approval	12 October, 2019	(1) Section 1.1 Identification (2) Section 1.2 Why is this User’s Guide needed?	(1) Added DOC-ID for SIS (2) Added Opaque File and Datatable definitions
Rev J	27 January, 2020	(1) Section 2.3 *MCWS Virtual Datatables and Opaque Files*	(1a) Added DataProductView (Opaque File) (1b) Added stream/FrameEvent

		(2) Section 4 *Accessing Realtime Content* (3) Section 4.1.1 *Creating a Stream from a Virtual Datatable* (4) Section 5 *Accessing Virtual Opaque File Content* (5) Section 8.9 *Virtual Datatable Definition – FrameEvent* (6) Section 8.10 *Virtual Datatable Definition – FrameSummaryEvent*	(2) Added note that an MCWS endpoint must be pinged once before a websocket connection can be established for the first time. (3a) Added FrameEvent to list of supported streams (3b) Deleted "Header channels (i.e. H-nnnn) are not provided over the Channel stream interface." (4) Describe DataProductView endpoint usage (5) New section (6) Changed bad_frames to bad_frame_count to match what is being published on the stream.
Rev J Approval	11 February, 2020	(1) Section 1.3 *Controlling Documents and Processes*	(1) Added MIMTaR Rev D and JPL SDR
Rev K	20 May, 2020	(1) Section 2.3 *MCWS Virtual Datatables and Opaque Files* (2) Section 3.2.1 *Performing a Query* (3) Section 3.4 *TSDB Queries* (4) Section 8.1 *Virtual Channel Definition – Channel*	(1) Added tlm/pass and tlm/tsdb for Channel, ChannelMinMax, EventRecord and Session. (1) Added Time Series Database (TSDB) column to table with Channel, ChannelMinMax, EventRecord and Session rows checked. (2) Added text specifying all ldb/qdb queries (except Session) require a session_id or time filter to avoid returning too much data. (3) New section

			(4) Modified "record_type" description to specify the realtime stream values. (4) Added note that name and module fields are not returned for AMPCS GLAD queries.
Rev K Approval	12 June, 2020	(1) Section 3.4 *TSDB Queries* (2) Section 8.1 *Virtual Channel Definition – Channel* (3) Section 8.4 *Virtual Channel Definition – CommandEvent* (4) Section 8.6 *Virtual Channel Definition – EventRecord* (5) Section 8.16 *Virtual Channel Definition – ChannelMinMax*	(1) Removed old idios that weren't true anymore. Added more examples since each data type has different mandatory filters. (2, 4, 5) Added text to scet and ert column descriptions that they are returned as epoch seconds for TSDB queries. (2) Added text to name and module column descriptions that they are not returned for lad queries (3) Added text to denote that request_id, dss_id, final and finalized fields are empty strings for realtime streams. (4) Added text to the description of the metadata_keywords and metadata_values fields are missing for realtime streams.
Rev L	19 September, 2020	(1) Section 1.3 *Controlling Documents and Processes* (2) Section 2.3 *MCWS Virtual Datatables and Opaque Files* (3) Section 3.3.1 *LAD Queries* (4) Section 8.1 *Virtual Channel Definition – Alarm* (5) Section 8.5 *Virtual Channel Definition – EventRecord*	(1) Updated SIS Rev F to G (2) Added stream/Alarm and lad/Alarm (3) Added LAD Alarm. Added text to describe that only LAD backend supported filters are applied to MCWS LAD queries. (4) New Section (5) Added text to denote that vcid is an empty string for realtime streams. Removed text from final, finalized and request_id that say these fields are empty for realtime. (6) Removed text from the description of the metadata_keywords and

		(6) Section 8.7 *Virtual Channel Definition – EventRecord*	metadata_values fields that these are missing from realtime streams.
Rev L Approval	11 November, 2020	(1) Section 1.2: *Why is this User's Guide Needed?* (2) Section 2.1: *Data Sources* (3) Section 2.3: *MCWS Virtual Datatables and Opaque Files*	(1) Added text describing MCWS capability to query historical and real-time telemetry. (2) Added a bullet point for describing the pass-through namespace. (3) Removed highlighting from footnote.

Table of Contents

1 Introduction*

1.1 Identification

This document, DOC-001496, is a supplement to the Mission Control Web Service (MCWS) RESTful SIS DOC-001489 Rev F that defines the operations and access interfaces for mission control data.

Element	MMCS (Multi-Mission Control Systems)
Subsystem	Mission Control
Assembly	MCWS
Title	Mission Control Web Service (MCWS) ReSTful API User's Guide
Configuration Number	820-061 631.11

1.2 Why is this User's Guide needed?

MCWS manages Mission Control information as Mission Information Objects (MIOs), including both Opaque File MIOs and Datatable MIOs. An Opaque File is defined as a collection of bits where there is no need for MCWS to understand what is inside the file. A Datatable is an organized, structured, parsable, and searchable collection of information organized by columns and rows.

When integrated with AMPCS for a mission adaptation, MCWS exposes Mission Control telemetry and telecommand information as Virtual Datatables. These Virtual Datatables do not physically exist – they are virtual constructs that perform like a user-defined view into the managed Telemetry information. However, they can be operated on just like real, physical Datatables.

This concept is used to perform historical queries with various query optimizations, to perform Latest Available Data (LAD) queries, and to receive streams of data as it is being processed in real time. To achieve this, MCWS can be configured to query from a variety of different data sources. For historical queries, MCWS can retrieve data from AMPCS databases, dictionary files, AMPCS Global LAD, Elasticsearch and Time Series DataBase (TSDB). For realtime streams, MCWS can listen to the AMPCS JMS message bus and provide data over websocket. In each of these cases, there is a unified HTTP query request syntax, standardized information models, and a standardized HTTP response and document structure.

At a technical implementation level, it is necessary to read and understand the content of the MCWS Software Interface Specification (SIS) in addition to this document.

1.3 Controlling Documents and Processes

Document	Release Date	Title
DOC-001455, Rev D	6/2019	MGSS Implementation and Maintenance Task Requirements, Rev. D
DOC-001489 Rev G	9/2020	MCWS L5 Software Interface Specification (SIS)

2 Telemetry Virtual Datatables

2.1 Data Sources

Telemetry data may be offered by more than one data source. For example, some missions incrementally export their complete Life of Mission (LOM) data set, and optimize the export for rapid query response. Some other telemetry information is only provided by AMPCS.

Different information sources may provide different qualities of service, or have different subsets of data, or both. For example:

- The "life of mission database" namespace (lom) provides highly optimized queries, but the data is not immediately available during realtime operations.
- As of MCWS 4.4, the "pass-through" namespace (pass) performs queries againt the Time Series DataBase (TSDB). This provides the data "as-is" in a format that adheres to MCWS expected output but supports limited filtering options due to its pass-through nature.
- The "load database" namespace (ldb) performs queries against the AMPCS load database, which is up to date with the realtime processing data flow, but may have slow query responses.
- The "query database" namespace (qdb) performs queries against the AMPCS query database, which is both complete and faster than the load database, but slower than the lom data source.

  Query responsiveness: In general, the lom data source (if implemented) will provide the best query responsiveness, but will be somewhat out of date. The ldb data source will provide the best query data latency, but will be slow to return data. The qdb data source is in between, in both of these dimensions.

- The "latest available data" namespace (lad) maintains only the most recently received value.
- The "realtime stream" namespace (stream) provides access to a stream of data, as it is being processed.
- The "snapshots" namespace (snapshots) is used to capture static snapshots of the other data sources.

MCWS formalizes the distinct data source concept, and the associated AMPCS-specific deployment architecture. The MCWS configuration defines the collection of registered namespaces, and thus what data sources are accessible. The registered namespace is the link to a particular data source (and for an AMPCS information provider, to the specific AMPCS deployment.)

2.2 Telemetry Query Concept

1. To access Telemetry information, users query a selected Virtual Datatable in a selected MCWS Registered Namespace, using the MCWS Datatable query syntax.
2. The Comon Access Manager (CAM) determines whether or not the user is allowed to access this particular Registered Namespace, using the CAM-provided SSO (Single Sign-On) token and the CAM-managed security policies.
3. Based on the Registered Namespace, MCWS identifies the Information Provider (an MCWS-internal proxy for the real data source) that manages the information.The Information Provider performs a query using the data-source-specific API. For example, the AMPCS Information Provider translates these queries into AMPCS-native queries and retrieves the information from the proper AMPCS instance based on project and venue.
4. The Information Provider further subsets the returned information (if necessary), and reformats it to comply with the Datatable response syntax.
5. Finally, the response content is returned to the user.

2.3 MCWS Virtual Datatables and Opaque Files

MCWS defines the following Telemetry Virtual Datatables and Virtual Opaque Files, and Command Event Virtual Datatable; see the appendix to this document for content definitions and details.

Note: the namespace: `gov/nasa/jpl/mcs/project/`*`projectName/venue/`*[2] is prepended to each item. Recommended venue names include: ops, atlo, testbed, and testset.

- Telemetry Engineering / Housekeeping / Analysis (EHA) channel information
  - `tlm/ldb/Channel`
  - `tlm/qdb/Channel`
  - `tlm/lad/Channel`
  - `tlm/lom/Channel`
  - `tlm/stream/Channel`
  - `tlm/tsdb/Channel`
  - `tlm/pass/Channel`
  - `tlm/ldb/ChannelDictionary/`*`dictionaryName`*
  - `tlm/ldb/ChannelEnumerationDictionary`

[2] For example, `gov/nasa/jpl/mcs/project/msl/ops/`

    - `/dictionaryName`
  - `tlm/lad/Alarm`
  - `tlm/stream/Alarm`
- Telemetry EHA channel information managed by Elastic Search or TSDB[3]
  - `tlm/lom/ChannelLAD`
  - `tlm/lom/ChannelMinMax`
  - `tlm/tsdb/ChannelMinMax`
  - `tlm/pass/ChannelMinMax`
- Data product information
  - `tlm/ldb/DataProduct`
  - `tlm/qdb/DataProduct`
  - `tlm/lom/DataProduct`
  - `tlm/stream/DataProduct`
  - `tlm/ldb/DataProductContent` (Opaque File)
  - `tlm/qdb/DataProductContent` (Opaque File)
  - `tlm/ldb/DataProductView (Opaque File)`
  - `tlm/qdb/DataProductView (Opaque File)`
- EVR information
  - `tlm/ldb/EventRecord`
  - `tlm/qdb/EventRecord`
  - `tlm/lad/EventRecord`
  - `tlm/lom/EventRecord`
  - `tlm/stream/EventRecord`
  - `tlm/tsdb/EventRecord`
  - `tlm/pass/EventRecord`
  - `tlm/ldb/EventRecordDictionary/`*`dictionaryName`*
- Other telemetry-related information
  - `tlm/ldb/Frame`
  - `tlm/qdb/Frame`
  - `tlm/lom/Frame`
  - `tlm/ldb/FrameContent` (Opaque File)
  - `tlm/qdb/FrameContent` (Opaque File)
  - `tlm/stream/FrameEvent`
  - `tlm/stream/FrameSummaryEvent`
  - `tlm/ldb/Packet`
  - `tlm/qdb/Packet`
  - `tlm/lom/Packet`
  - `tlm/ldb/PacketContent` (Opaque File)
  - `tlm/qdb/PacketContent` (Opaque File)
  - `tlm/stream/PacketSummaryEvent`
- Command Event information
  - `tlm/ldb/CommandEvent`
  - `tlm/qdb/CommandEvent`

---

[3] These Datatables do not contain the same structure as the Channel Datatable; see the appendix for definitions.

    - `tlm/stream/CommandEvent`
    - `tlm/lom/CommandEvent`
- AMPCS-specific information
    - `tlm/ldb/Session`
    - `tlm/qdb/Session`
    - `tlm/lad/Session`
    - `tlm/lom/Session`
    - `tlm/tsdb/Session`
    - `tlm/pass/Session`
    - `tlm/stream/Message`
- Unconstrained information[4] may be saved and retrieved from the snapshots namespace
    - `tlm/snapshots`

Users and user application can access these Virtual Datatables in accordance with the following access patterns:

Virtual Datatable Name	Historical Query (ldb)	Historical Query (qdb)	Life of Mission Query (lom)	(AMPCS) Latest Available Data (lad)	(AMPCS) Streaming Data (stream)	(Elastic Search)	Time Series Database (TSDB)
Alarm				☑	☑		
Channel	☑	☑	☑	☑	☑		☑
ChannelLAD						☑	
ChannelMinMax						☑	☑
ChannelDictionary	☑						
ChannelEnumeration Dictionary	☑						
CommandEvent	☑	☑	☑		☑		
DataProduct	☑	☑	☑		☑		
EventRecord	☑	☑	☑	☑	☑		☑
EventRecordDictionary	☑						
Frame	☑	☑	☑				
FrameEvent					☑		
FrameSummaryEvent					☑		
Message					☑		
Packet	☑	☑	☑				
PacketSummaryEvent					☑		
Session	☑	☑	☑	special meaning			☑

[4] Using any desired sub-namespace structure, and managed as actual (not virtual) Datatables or Opaque files

Users and user applications can access these Virtual Opaque Files in accordance with the following access patterns:

Virtual Opaque File Name	Historical Query (ldb)	Historical Query (qdb)	Life of Mission Query (lom)	(AMPCS) Latest Available Data (lad)	(AMPCS) Streaming Data (stream)	(Elastic Search)
DataProductContent	☑	☑				
DataProductView	☑	☑				
FrameContent	☑	☑				
PacketContent	☑	☑				

The above information is generally[5] converted to URLs as follows:

Full URL:
`https://<server>/<webapp><project><venue><resource><parameters>`

Example values:

- `<server>` : `websrv:8443`
- `<webapp>` : `mcws`
- `<project>` : `/gov/jpl/nasa/mcs/project/msl`
- `<venue>` : `/ops`
- `<resource>` : `/tlm/lom/Channel`
- `<parameters>` : `?filter=(channel_id=(P-1200),scet>2016-123T00:00:00`

[5] actual URLs are deployment specific, and mission GDSE managed.

3 Accessing Virtual Datatable Content

MCWS Virtual Datatables provide read-only access to telemetry information, through the pre-defined Datatables described in this document. To retrieve data, the user creates a request filter that describes the information of interest, and then requests that the information be retrieved and returned.

3.1 Defining Filters

Filters are defined in terms of Virtual Datatable column names. Column names are defined in Section 7.

3.1.1 Identifying data columns

Use the `select=()` filter to define the columns to be returned.

The `select=()` option is used to establish the specific columns of interest, and their data return order.

- If this option is not provided, all columns will be returned, in the order defined in the Virtual Datatable definitions in this document.
- Otherwise, the desired column names are listed, comma separated, inside the select parentheses. The order the column names are listed will define the order in the output.
- It is an error to define a select() filter for a column name that is not part of the Virtual Datatable.

Example: the user chooses a particular subset of the available columns in the EventRecord Virtual Datatable:

```
select=(event_id,name,level,module,scet,message)
```

3.1.2 Identifying data rows

Use the `filter=()` filter to identify the rows to be returned.

The filter=() option is used to establish the specific rows of interest.

- Use of a valid filter=() is mandatory.
- Filters are created by constructing rules.
    - A rule is composed of a column name, a comparison operator, and a comparison value.

    - Comparison operators are “=”, “!=”, “<”, “<=”, “>=”, “>”[6].
    - Comparison values are arbitrary, user-supplied values, or lists of values inside a parenthesis pair. If a list of values is supplied, then any item in the list can be used to satisfy the filter. Wildcards (* and %) may be used to specify values.
- The desired rules are listed, comma separated, inside the filter parentheses; if more than one rule is listed, then all rules must be simultaneously satisfied for a row to be considered to have passed the filter.
- It is an error to define a filter for a column name that is not part of the Virtual Datatable.
- It is not an error to define a filter that results in 0 returned rows.

3.1.3 Sorting returned rows

This filtering option is only supported for historical queries (lom, ldb, qdb.) If supplied, it is ignored for queries on other query types.

Use the `sort=()` filter to identify result sorting criteria.

The sort=() option is used to describe the columns by which the filtered rows should be sorted before returning the content.

- If this option is not provided, no sort order is defined; the underlying information source may provide a default sort option.
- Otherwise, the desired column names are listed, comma separated, inside the sort parentheses. Sort qualifiers (such as numeric, descending, and unique) can be optionally provided. The order the column names are listed, and their qualifiers, will determine the order in which the filtered rows will be returned.
- The user may choose to sort on columns that are not part of the filtered output.
- It is an error to define a sort() filter for a column name that is not part of the Virtual Datatable.
- Sorting large return data sets will result in poor query performance. Sorting very large return data sets may result in exhaustion of MCWS server resources (such as temp disk space.)

---

[6] Note that the actual behavior of “<” and “<=” , and “>” and “>=”, is data source specific.

3.2 Historical Queries

3.2.1 Performing a Query

A Historical Query is performed by issuing a GET request against the Virtual Datatable of interest, using a mandatory filter=() parameter and optional select=() and sort=() parameters.

Pretty-printed example:

```
GET .../testbed/tlm/ldb/Channel?
filter=(
 session_id=2306,
 type!=(STATUS,BOOLEAN),
 channel_id=(P-*)
)
&select=(channel_id,ert,scet,dn,eu)
&sort=(ert)
```

Where available, a query against the exported LOM data can be performed using the exact same query syntax (though using query parameters that are more appropriate for a data source that spans many sessions), but against an alternate resource:

```
GET .../ops/tlm/lom/Channel?
filter=(
 lst=SOL-0514,
 type!=(STATUS,BOOLEAN),
 channel_id=(P-*)
)
&select=(channel_id,ert,scet,dn,eu)
&sort=(ert)
```

All AMPCS Load or Query Database (i.e., ldb or qdb) queries, except Session, require a session_id or time filter to avoid returning too much data. Note that for data queried from those sources, MCWS will escape all double-quote and backslash characters.

3.2.2 Performing a Dictionary Query

The list of available dictionary names can be found using a metadata query on the dictionary resource.

A metadata query uses a trailing "/" to retrieve metadata about a particular resource, instead of data from that resource.

Pretty-printed example:

```
GET .../atlo/tlm/ldb/ChannelDictionary/
```

A Dictionary content query follows the same syntax as a Historical query, except that the name of the dictionary may be provided.

Pretty-printed example:

```
GET .../atlo/tlm/ldb/ChannelDictionary/chan_dict_
v1-0
```

If the name of the dictionary is not provided, the default AMPCS "CURRENT" dictionary is assumed.

Pretty-printed example:

```
GET .../ops/tlm/ldb/ChannelDictionary
     ?filter=(channel_id=P-1034)
```

3.3 LAD Queries

The different data sources for LAD (Elastic Search vs AMPCS native) provide different Virtual Datatables with different column definitions. See section 7 for the specific column definitions.

3.3.1 LAD Queries (Alarm, Channel or EVR Virtual Datatable)

A Latest Available Data (LAD) Query is performed by issuing a GET request against the lad/Alarm, lad/Channel or lad/EventRecord Virtual Datatable, using a mandatory filter=() parameter and optional select=() parameters.

- The desired LAD time system is identified by supplying the mandatory parameter lad_type=*LADTypeValue*, where *ladTypeValue* is one of "ert", "scet", or "event"
- MCWS LAD Queries do not apply any filtering within MCWS; only filters that are supported by the backend are applied. Other filters will be silently ignored.
    - Channel and Alarm requests support time filters, "session_id" and "channel_id".
    - Event Record requests support time filters, "session_id", "event_id", "level" and "name".

Pretty-printed example:

```
GET .../atlo/tlm/lad/Channel
     ?lad_type=ert,
     &filter=(channel_id=(A-1234,B-5678))
```

3.3.2 LAD Queries (ChannelLAD Virtual Datatable, Elastic Search)

A Latest Available Data (LAD) Query against the Elastic Search datastore is performed by issuing a GET request against the ChannelLAD Virtual Datatable, using a mandatory filter=() parameter and optional select=() parameters. Note that the columns that can be returned are different than those managed by the Channel Virtual Datatable.

- The desired LAD time system is identified by supplying the mandatory parameter lad_type=*LADTypeValue*, where *ladTypeValue* is one of "ert", "scet"

Pretty-printed example:

```
GET .../atlo/tlm/lom/ChannelLAD
     ?lad_type=scet,
     &filter=(channel_id=(A-1234,B-5678))
     &select=(scet,ert,eu_or_dn)
```

3.3.3 LAD Queries (Session Virtual Datatables)

Performing a LAD query for Session data returns information about sessions that are currently active[7].

Pretty-printed example:

```
GET .../atlo/tlm/lad/Session
```

3.4 TSDB Queries

A Time Series Database pass-through query against the TSDB datastore is performed by issuing a GET request against the Virtual Datatable of choice. Mandatory parameters vary per data type. Since this is a pass-through query, output is always returned in JSON. The `select`, `sort` and `output` parameters have no effect on the query. Results are returned already sorted by the scet time type.

Pretty-printed Session example (no mandatory parameters):

```
GET .../atlo/tlm/pass/Session
```

Pretty-printed Channel example (mandatory `filter` parameter for start/end scet and channel_id):

```
GET .../atlo/tlm/pass/Channel
     ?filter=(scet<=2021-201T00:00:00,
     scet>=2019-201T00:00:00,
     channel_id=(A-1234,B-5678))
```

Pretty-printed EventRecord example (mandatory `filter` parameter for start/end scet):

```
GET .../atlo/tlm/pass/EventRecord
     ?filter=(scet<=2021-201T00:00:00,
     scet>=2019-201T00:00:00,
```

[7] Active sessions are defined as sessions that have an active downlink processor associated, whether or not data is currently flowing.

```
channel_id=(A-1234,B-5678))
```

Pretty-printed ChannelMinMax example (mandatory `filter` parameter for start/end scet and `minmax` parameter):

```
GET .../testset/tlm/pass/ChannelMinMax
     ?filter=(scet>2015-201T00:00:00,
     scet<2015-202T00:00:00,
     channel_id=(GPSP-0000)
     &minmax=(3,scet,123)
```

In this example, the user wants to retrieve minmax data that has been binned into 3 bins, each equally dividing the time range of interest (all of scet day 2015-201). The last field of the `minmax` parameter is only required to satisfy the interface but has no effect on the query.

4 Accessing Realtime Content

A realtime, streaming query has a more complex use model than the other query types. In contrast to the Historical, LAD, and Dictionary queries, a streaming query establishes a stateful interaction channel (a "stream") between a client and the MCWS. Information received in realtime, and that matches the specified filter criteria, will be sent to the client while the stream is established.

There are several considerations when performing streaming queries:

- An MCWS endpoint (other than a stream) must be pinged at least once before the very first streaming query can be established after MCWS startup.
- Each stream has an associated collection of columns and rows provided by the underlying Datatable, established by specifying a select() and filter() parameter when the stream is created.
- Creating a stream establishes a cache on the server, and thus consumes resources. Because of this, streams should be managed by the user, and also must be managed by the server.
    - The user must close a stream when it is no longer needed.
    - The server can close a stream, at any time. If the used cache size has reached a defined size limit, the server will close the stream and reclaim those resources.
- Once created, a stream's definition is immutable.
- A new stream starts caching data only after a client connection over the websocket has been established.
- In contrast to historical queries, filters on streams should not use time-based filters.

4.1 Creating a Stream

Creating a stream requires the use of an HTML-5 compliant websocket. The websocket URL follows the typical pattern:

```
wss://server:port/app/resource
     ?select=()
     &filter=()
```

4.1.1 Creating a Stream from a Virtual Datatable

Only Alarm, Channel, DataProduct, EventRecord, FrameEvent, FrameSummaryEvent, Message, PacketSummaryEvent, and CommandEvent Virtual Datatables support streams.

The URL follows the following pattern:

```
wss://server:port/.../ops/tlm/stream/Channel
     ?select=(channel_id,ert,scet,dn,eu,status)
```

```
&filter=(channel_id=(P-1200,M-010),
topic=topic_name, session_id=1126)
```

Note that the "topic" filter parameter is mandatory, and can be retrieved from the Session LAD Virtual Datatable. Specifying a session_id is optional.

Note the following limitations:

- For Channel streams, a filter containing "channel_id" is mandatory.
- For EventRecord streams, a filter containing "module" is mandatory.

4.1.2 Creating a Stream from a Datatable

Note: this feature is primarily used for non-operational uses such as demonstrations and client testing.

This feature combines the interface characteristics of a realtime stream, but uses a Datatable instead of the realtime message bus as the source of information. Any valid Datatable can be used as a source of data to be sent over the websocket interface.

The resource URL follows the following pattern:

```
wss://server:port/.../any/datatable
     ?select=(channel_id,ert,scet,dn,eu,status)
     &filter=(channel_id=(P-1101,M-011))
```

The request can also include the optional `meter` parameter:

`meter=none` (or the parameter is not supplied) : meter the data at the SCET-based data rate, if one can be determined, which results in data that is returned at a rate consistent with the time-based separation between rows.

`meter=N` : meter the data at a rate of N rows per second

4.2 Reading from a Stream

Stream content is accessed through the websocket protocol, and provided in JSON Datatable format, as defined by the select() parameter.

It is the client's responsibility to rapidly read from the websocket connection. The server provides Virtual Datatable rows, in JSON format only. Only data that passes the filter() parameter will be returned to the stream's reader. Returned data is not sorted.

Note that for realtime streaming data, MCWS will escape all double-quote and backslash characters.

4.3 Writing to a Stream

While the websocket protocol is bidirectional, no write functions are supported. Written data will not be read by the server.

4.4 Disconnecting from a Stream

It is the client's responsibility to close the connected websocket when no longer needed. Note also that the server may terminate a stream, if it determines that the connected client is reading streaming data too slowly, and can thus no longer cache[8] streaming data on behalf of the user.

[8] This is the "punish slow subscribers with a disconnect" model.

5 Accessing Virtual Opaque File Content

The FrameContent and PacketContent Virtual Opaque Files generate and return a file of binary content, as if the content had been stored as an Opaque File. All binary content that matches the filter criteria (consistent with the Virtual Datatable) is concatenated and returned to the user. A time sort filter should be used, but is optional.

Pretty-printed example:

```
GET .../ops/tlm/qdb/PacketContent
     ?filter=(apid=20,
       scet>2015-201T00:00:00,
       scet<2015-201T00:15:00)
     &sort=(scet)
```

The DataProductContent Virtual Opaque File also generates and returns a file of binary content.

The DataProductView Virtual Opaque File processes the product file according to its telemetry dictionary definition and will return a plain text file.

A filter parameter is required for both of these endpoints. The filter parameter requires that unique_name be specified, where unique_name is the absolute file path without the extension. The Data Product's mandatory unique_name value can be found via query (qdb, ldb, or stream) to the DataProduct Virtual Datatable. DataProductContent also requires the filter parameter to supply the session_id. filetype is an optional filter parameter for which .dat, .pdat, .emd, and .pemd are the possible values; if no value is supplied, .dat is assumed.

Pretty-printed examples:

```
GET .../ops/tlm/qdb/DataProductContent
     ?filter=(session_id=96,unique_name=/Users/…/
     McamLThumbnail_0457586850-65432-1)
     &filetype=.emd

GET .../ops/tlm/qdb/DataProductView
     ?filter=(unique_name=/Users/…/McamLThumbnail
     _0457586850-65432-1)
     &filetype=.pdat
```

The DataProductContent and DataProductView endpoints do not use a database call in the backend so the product files must be stored on the same host as MCWS. If the AMPCS host is different than the MCWS host, projects can use a shared NFS mount to cross mount product files or use a remote LDI archiver.

6 Accessing Aggregated Content

6.1 Channel MinMax information

The ChannelMinMax virtual datatable provides access to subsetted information[9] over a particular time range.

Note that the Datatable `minmax` parameter provides an API to a generalized minmax function, but the Elastic Search ChannelMinMax Virtual Datatable only implements one value column – the "eu_or_dn" column[10].

Note that no information about the size of the bin, or the number of data points in the bin, is available through this virtual Datatable. There is also no indication of whether a returned row is a min value or a max value.

Pretty-printed example:

```
GET .../ops/tlm/lom/ChannelMinMax
     ?filter=(channel_id=(A-1234,B-5678),
       scet>2015-201T00:00:00,
       scet<2015-202T00:00:00)
     &select=(scet,ert,eu_or_dn)
     &minmax=(256,scet,eu_or_dn)
```

In this example, the user wants to retrieve minmax data that has been binned into 256 bins, each equally dividing the time range of interest (all of scet day 2015-201). The minmax algorithm will return the minimum "eu_or_dn" value and maximum "eu_or_dn" in each of those bins, resulting in up to 512 data rows being returned for each channel. Where no value is present in a bin, no min or max row will be returned. The columns returned are specified by the `select=()` parameter, consistent with other uses of the MCWS.

---

[9] This feature is specifically in support of an interactive display tool, where the desired number of results and spanning time range is under the user's control.

[10] In the Elastic Search implementation, this is the "y" column.

7 Accessing Dictionary Content

7.1 AMPCS Channel and EVR Dictionary Information

AMPCS Channel and EVR dictionary information is available via MCWS.

Pretty-printed example:

```
GET .../ops/tlm/qdb/ChannelDictionary
```

or

```
GET …/ops/tlm/qdb/ChannelDictionary/
R12_0_3_20140813_02
```

MCWS uses AMPCS tools to retrieve the mission dictionary information and then stores the results as a flat file. In order to enhance the performance of returning the dictionary information, subsequent requests for the information are performed using the flat files instead of the AMPCS tools. A file will be used for 24 hours, after which the next query will be used to create a new flat file. If the dictionary information must be updated before the 24-hour expiration period elapses, the cache must be cleared so that a new flat file can be created. Note that the flat files persist between MCWS server startup events.

Pretty-printed example:

```
GET .../ops/tlm/qdb/ChannelDictionary?clear_cache
```

A list of the dictionaries available at the specified endpoint can be retrieved.

Pretty-printed example:

```
GET .../ops/tlm/qdb/ChannelDictionary/
```

8 Acknowledgment

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

9 Appendix: Virtual Datatable Definitions

9.1 Virtual Datatable Definition – Alarm

The Alarm datatable definition is essentially the same as the Channel datatable definition. The differences are documented here; refer to the Channel definition for the full list of column names.

Column Name	Description
record_type	Value is always the literal string 'alarm'.
alarm_change	Indicates whether this channel just entered alarm, is still in alarm or just went out of alarm. Value can be “Entered Alarm”, “Still in Alarm” or “Exited Alarm”. String.

9.2 Virtual Datatable Definition – Channel

Column Name	Description
record_type	Value is always the literal string 'eha' with the exception of realtime streams. *AMPCS R7 realtime stream*: ‘EhaChannel’ *AMPCS R8 realtime stream*: ‘AlarmedEhaChannel’
session_id	The AMPCS session ID. Positive integer greater than zero.
session_host	The name of the host on which the session was run. String, maximum 64 characters.
channel_id	The channel identifier from the telemetry dictionary. String, maximum 16 characters.
dss_id	The receiving station ID for this channel sample. Populated only for FSW, header, and monitor channels. Unsigned integer.
vcid	Virtual channel id. Populated only for FSW and header channels. Unsigned integer.
name	The long name of the channel from the telemetry dictionary. String, maximum 64 characters. Not returned for lad queries because the AMPCS Global LAD does not store all static fields.
module	The name of the software module generating this channel, from the telemetry dictionary. String, maximum 32 characters. Not returned for lad queries because the AMPCS Global LAD does not store all static fields.
ert	The Earth Receive Time for the channel sample. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings.

	For TSDB queries, time is returned as epoch seconds. Unsigned integer.
scet	The Spacecraft Event Time of the channel sample. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings. For TSDB queries, time is returned as epoch seconds. Unsigned integer.
lst	The Local Solar Time of the channel sample. String in the format SOL-XXXXMHH:mm:ss.SSS.
sclk	The Spacecraft Clock Time of the channel sample. String of the format CCCCCCCCCC-FFF[FF] or CCCCCCCCCC.FFF[?FF], depending on project settings.
dn	The channel sample Data Number. This is the raw value received from the spacecraft or SSE. Data type is channel dependent.
eu	For numeric channels, this is the floating point Engineering Unit value for the channel, if an EU is defined. Empty otherwise.
status	For STATUS and BOOLEAN channels, this is the string state that the DN maps to.
dn_alarm_state	Alarm status, for alarms computed on the Data Number. Value can be 'RED', 'YELLOW', or 'NONE'.
eu_alarm_state	Alarm status, for alarms computed on the Engineering Unit. Value can be 'RED', 'YELLOW', or “NONE”.
realtime	Flag indicating whether the channel sample is from realtime or recorded telemetry. Value can be 'true' or 'false'.
type	The data type of the channel. Value can be 'SIGNED_INT', 'UNSIGNED_INT', 'DIGITAL', 'STATUS', 'FLOAT', 'BOOLEAN', 'ASCII', or 'UNKNOWN'.

9.3 Virtual Datatable Definition – ChannelDictionary

Column Name	Description
record_type	Value is always the literal string 'channel_dictionary'.
channel_id	Channel Identification. String. The short name of the channel.
channel_name	Channel Name. String. The long name of the channel.
data_type	Channel Data Type. String. The data type of the DN. (Note that the data type of the EU is always Float.)
dn_to_eu	DN to EU Flag. Boolean, true or false. Whether or not a EU conversion should be performed for this channel.
dn_units	DN Value Units. String. The units of the DN
eu_units	EU Value Units. String. The units of the EU.
channel_format	The intended display format of the channel DN value. Values adhere to the C printf standard for format strings.
io_format	Input/output format of the DN channel value as specified in the Channel Dictionary. String.
eu_io_format	Input/output format of an EU channel value as specified in the Channel Dictionary. String.

9.4 Virtual Datatable Definition – ChannelEnumerationDictionary

Column Name	Description
channel_id	The channel identifier from the telemetry dictionary. String, maximum 16 characters.
channel_type	String, BOOL, or ENUM.
channel_format	The intended display format of the channel DN value. Values adhere to the C printf standard for format strings.
enum_value	Enumerated channel value. Signed integer.
enum_string	Enumerated channel string. String.

9.5 Virtual Datatable Definition - CommandEvent

Column Name	Description
record_type	Value is always the literal string 'command_event'.
session_id	The AMPCS session ID. Positive integer greater than zero.
session_host	The name of the host on which the session was run. String, maximum 64 characters.
event_time	UTC string in the format YYYY-MM-DDThh:mm:ss.
request_id	Identifier for command request. String.
command_type	The type of command message. Value can be "UNKNOWN","FlightSoftwareCommand", "HardwareCommand", "SseCommand", "FileLoad", "SCMF", "RawUplinkData", "SequenceDirective".
status	The status of the command request. Value can be "UNKNOWN", "Requested", "Scheduled", "Radiating", "Radiated", "Radiation_Error", "Radiation_Failed", "Deleted", "Expired_Window", "Windows_Expired", "Rad_Attempts_Exceeded", "Received", "Corrupted", "Aborted", "Not_Radiated", "Awaiting_Confirmation".
command_string	The string representation of the command.
scmf_file	The full path and filename of the generated (or archived) SCMF.
original_file	The full path and filename of the original SCMF, raw data or other file.
fail_reason	If a command fails to transmit the reason is captured as a string.
checksum	Checksum of the command message. Unsigned integer.
total_cltus	The number of CLTUs associated with the command. Unsigned integer.
dss_id	The transmitting station ID for the command message. Unsigned integer. Empty string for realtime streams.
bit1_rad_time	The time the first bit of the command was radiated. UTC string in the format YYYY-DDDTHH:MM:SS.FFF.
last_bit_rad_time	The time the last bit of the command was radiated. UTC string in the format YYYY-DDDTHH:MM:SS.FFF.
final	Boolean, true or false. Multiple command request status values are final (e.g., "Radiated"). Value is "TRUE" if command status value is one of the final status values.
finalized	Boolean, true or false. Value is "TRUE" if command has a "final" value equal to "TRUE". Once "finalized" is set to "TRUE", the value of "status" will not change.
vcid	Virtual channel id. Unsigned integer. Empty string for realtime streams.
commanded_side	Used to indicate the processor for which the command was intended, as in Mars Rover Compute Element (RCE) A or B. String. Default value is an empty string.

9.6 Virtual Datatable Definition – DataProduct

Column Name	Description
record_type	Value is the literal string 'product'. For realtime streams value will be “product_started”, “complete_product”, “product_part_received” or “partial_product”.
session_id	The AMPCS session ID. Positive integer greater than zero.
session_host	The session name. String, maximum of 64 characters.
vcid	The virtual channel ID on which the product was received. Unsigned integer.
apid	The product Application Process ID from the telemetry dictionary. Unsigned integer.
product_type	The type of the product (APID name). String
creation_time	The time that AMPCS wrote the data product on the ground. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings.
scet	The Data Validity Time (DVT) of the product, as a Spacecraft Event Time. UTC string in the format YYYY-MMDDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings..
lst	The Data Validity Time (DVT) of the product, as a Local Solar Time. String in the format SOL-XXXXMHH:mm:ss.SSS.
ert	The Earth Receive Time for the product, based upon the ERT of the first product packet received. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings..
sclk	The Data Validity Time (DVT) of the product, as a Spacecraft Clock Time. String of the format CCCCCCCCCC-FFF[FF] or CCCCCCCCCC.FFF[FF], depending on project settings.
command_number	The serial number in the sequence file, starting with one, of the command that triggered product generation. Unsigned integer.
dvt_coarse	The coarse SCLK portion of the Data Validity Time in seconds/ticks. Unsigned integer.
dvt_fine	The fine SCLK portion of the Data Validity Time, in sub-seconds/ticks. Unsigned integer.
total_parts	Total number of data packets in the product. Unsigned integer.
seq_id	Sequence id. Unsigned integer.
seq_version	The version of the sequence containing the command that generated the product. Unsigned integer.
file_size	Size of the data product file in bytes. Unsigned integer.
checksum	Checksum of the data product file. Unsigned integer.
ground_status	Status of product. Value will be “UNKNOWN”, “PARTIAL”, “COMPLETE”, “COMPLETE_NO_CHECKSUM”, “COMPLETE_CHECKSUM_PASS”, or “PARTIAL_CHECKSUM_FAIL”. String.
version	Version of product. Decimal (like 10.101).
unique_name	The absolute path associated with the data file.

Below fields are only present in realtime data streams	
transaction_id	The transaction number (for example CFDP transaction number). Only populated in realtime data; will be an empty string in historical. Unsigned integer.
part_number	The number of this part within the product. Note that whether the part number starts at 0 or 1 is mission specific. Only present for realtime "product_part_received" record types. Unsigned integer.
reason	The reason for triggering data product assembly. Value will be "NO_TRIGGER", "END_PART", "AGING_TIMER", "PROD_CHANGE", "END_DATA", "END_TEST", "DICTIONARY_CORRECTION", or "CFDP_AMPCS_PRODUCT_PLUGIN". Only present for realtime "partial_product" record types. String.
log	Transaction log file path and name. Only present for realtime "partial_product" record types. String.
creation_time	The product ground creation time as an ISO formatted string YYYY-DOYTHH:mm:ss.SSS. Only populated for "partial_product" and "complete_product" record types. Will be an empty string for "product started" and "product_part_received" record types.
event_time	The message event time (wall clock time) as YYYY-DOYTHH:mm:ss.SSS.

Realtime stream "product_started" record types do not have associated metadata so only the following fields are expected to have data. All others will be an empty string.

- record_type
- session_id
- session_host
- vcid
- apid
- total_parts
- transaction_id
- event_time

9.7 Virtual Datatable Definition – EventRecord

Column Name	Description
record_type	Value is always the literal string 'evr'.
session_id	The AMPCS session ID. Positive integer.
session_host	The name of the host on which the session was run. String, maximum of 64 characters.
name	The telemetry dictionary name of the EVR. String, maximum of 128 characters.
module	The name of the software module generating this EVR, from the telemetry dictionary. String, maximum of 32 characters.
level	The level of the EVR from the telemetry dictionary. String, maximum 16 characters.
event_id	The numeric identifier of the EVR, from the telemetry dictionary. Unsigned integer.
vcid	Virtual channel id. Unsigned integer.
dss_id	Receiving station id. Unsigned integer.
from_sse	Flag indicating that the EVR is from the Simulation Support System (SSE) as opposed to the flight system. Value can be 'true' or 'false'.
realtime	Flag indicating whether the EVR is from real time or recorded telemetry. Value can be 'true' or 'false'.
sclk	The Spacecraft Clock Time of the EVR. String of the format CCCCCCCCCC-FFF[FF] or CCCCCCCCCC.FFF[?FF], depending on project settings.
scet	The Spacecraft Event Time of the EVR. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings. For TSDB queries, time is returned as epoch seconds. Unsigned integer.
lst	The Local Solar Time of the EVR. String in the format SOL-XXXXMHH:mm:ss.SSS.
ert	The Earth Receive Time for the EVR. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings. For TSDB queries, time is returned as epoch seconds. Unsigned integer.
rct	The AMPCS Record Creation Time of the EVR record. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings.
message	The formatted EVR message. String, maximum of 383 characters.
metadata_keywords	List of metadata keywords associated with this EVR. String of the form '[(keyword1),(keyword2),(keyword3)]'. The value '[]' indicates there was no metadata.
metadata_values	List of metadata values found in the EVR, in the same order as the keyword list. String of the form '[(value1),(value2),(value3)]'. The value '[]' indicates there was no metadata.

9.8 Virtual Datatable Definition – EventRecordDictionary

Column Name	Description
record_type	Value is always the literal string 'evr_dictionary'.
evr_id	EVR Identification. String. The short name of the EVR.
evr_name	EVR Name. String. The long name of the EVR.
level	EVR Level. String. The level (FATAL, WARN, INFO, etc) of the EVR.
module	FSW Module. String. The module associated with the EVR.
ops_cat	Ops Category. String. The operations category associated with the EVR.
nargs	Number of arguments in the format string. Integer.
format	EVR Format string. String. The “printf-style” formatting string used to produce the EVR content.

9.9 Virtual Datatable Definition – Frame

Column Name	Description
record_type	Value is always the literal string 'frame'.
session_id	The AMPCS session ID. Positive integer greater than zero.
session_host	The name of the host on which the session was run. String, maximum of 64 characters.
type	The type of the telemetry frame. String, maximum of 32 characters. Values are project dependent.
ert	The Earth Receive Time for the frame. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings.
rct	The record creation time for the frame. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings.
relay_spacecraft_id	Numeric spacecraft ID of the relay spacecraft through which the frame was transmitted. Unsigned integer.
vcid	Virtual Channel ID on which the frame was received. Unsigned integer.
vcfc	Virtual Channel Frame Counter for the frame. Unsigned integer.
dss_id	Receiving station ID. Unsigned integer.
bit_rate	Data rate at which the frame data was received, in bits/second. Floating point number.
is_bad	Flag indicating whether the frame was invalid. Value is 'true' or 'false'.
bad_reason	Reason the frame was bad/invalid, if it was. Values can be 'UNKNOWN', RS_ERROR', 'CRC_ERROR', 'BAD_VERSION', 'BAD_SCID', 'BAD_HEADER', 'BAD_VCID', 'BAD_PKT_POINTER', 'TURBO_ERROR', 'UNKNOWN_VCID'
length	Length in bytes of the frame. This includes the CCSDS frame header. If the --restoreBodies option is supplied, this length will be adjusted to account for any frame headers or trailers re-attached to the frame body. Unsigned integer.

9.10 Virtual Datatable Definition – FrameEvent

Column Name	Description
record_type	Value is always the literal string 'frame_event'.
session_id	The AMPCS session ID. Positive integer greater than zero.
session_host	The name of the host on which the session was run. String, maximum of 64 characters.
message_type	The type of the telemetry frame. Value will be one of 'OutOfSyncBytes' (for AMPCS R7) or 'OutOfSyncData' (for AMPCS R8), 'InSync', 'BadFrame' (for AMPCS R7) or 'BadTelemetryFrame' (for AMPCS R8), 'LossOfSync' or 'FrameSequenceAnomaly'. String.
summary	Message summary that will vary according to the message type. String.
ert_time	The Earth Receive Time for the frame. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings.
event_time	The message event time (wall clock time) as YYYY-DOYTHH:mm:ss.SSS.
bit_rate	Data rate at which the frame data was received, in bits/second. Floating point number.
frame_type	The dictionary transfer frame type as defined by the AMPCS transfer_frame.xml. Empty string when message_type is OutOfSync. String.
dss_id	Receiving station ID. Unsigned integer.
vcid	Virtual Channel ID on which the frame was received. Empty string when message_type is OutOfSync. Unsigned integer.
scid	Numeric spacecraft ID. Empty string when message_type is OutOfSync. Unsigned integer.
reason	Reason the frame was bad/invalid, if it was. Populated only when message_type is BadFrame. Values can be 'UNKNOWN', RS_ERROR', 'CRC_ERROR', 'BAD_VERSION', 'BAD_SCID', 'BAD_HEADER', 'BAD_VCID', 'BAD_PKT_POINTER', 'TURBO_ERROR', 'UNKNOWN_VCID'.
actual_vcfc	The actual frame counter. Populated only when message_type is FrameSequenceAnomaly. Unsigned integer.
expected_vcfc	The expected frame counter. Populated only when message_type is FrameSequenceAnomaly. Unsigned integer.

9.11 Virtual Datatable Definition – FrameSummaryEvent

Column Name	Description
record_type	Value is always the literal string 'frame_summary_event'.
session_id	The AMPCS session ID. Positive integer greater than zero.
event_time	The time AMPCS generated the packet summary message (wall clock time). String in the format YYYY-DOYTHH:MM:SS.FFF.
dead_frames	The number of dead frames received. These are considering in-sync frames but they are not processed. Unsigned integer.
frame_bytes	The number of in-sync frame bytes processed so far (data + idle frame byte count). Unsigned integer.
idle_frames	The number of idle (non-data) frames received so far. These are considered in-sync frames but they are not processed. Unsigned integer.
insync	Flag indicating whether the frame synchronizer is currently in lock state. Value is 'true' or 'false'.
num_frames	The number of data + idle (in-sync) frames seen so far. Unsigned integer.
out_of_sync_bytes	The number of out-of-sync bytes seen so far. Unsigned integer.
bad_frame_count	The number of bad (invalid) frames seen so far. Unsigned integer.
out_of_sync_count	The number of times out-of-sync events have occurred so far. Unsigned integer.
frame_summaries	Contains an entry for each frame type/VCID combination. Array.
	vcid: Virtual Channel ID on which the frame was received. Unsigned integer.
	frame_type: Type of frame being transmitted; usually defined in the FGICD. String.
	last_ert_time: The Earth Receive Time of the latest frame received for the selected virtual channel and frame type. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings.
	sequence_count: Virtual Channel Frame Counter (VCFC). Unsigned integer.
	instance_count: The number of frames received for the frame type/VCID combination. Unsigned integer.
encoding_summaries	Contains an entry for each frame encoding/VCID combination. Array.
	vcid: Virtual Channel ID on which the frame was received. Unsigned integer.
	encoding_type: Encoding type (TURBO, Reed-Solomon, etc). String.
	last_ert_time: The Earth Receive Time of the latest frame received for the selected virtual channel and encoding type. UTC

	string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings.
	sequence_count: Virtual Channel Frame Counter (VCFC). Unsigned integer.
	instance_count: The number of frames received for the encoding type/VCID combination. Unsigned integer.
	bad_frame_count: The total number of invalid frames received for this encoding type/VCID combination. Unsigned integer.
	error_count: The total number of encoding errors in frames for this encoding type/VCID combination. Unsigned integer.

9.12 Virtual Datatable Definition – Message

Column Name	Description
record_type	Value is always the literal string 'message'.
session_id	The AMPCS session ID. Positive integer.
session_host	The name of the host on which the session was run. String, maximum of 63 characters.
message_type	*AMPCS R7* valid values are 'Heartbeat', 'RawDataSummary', 'StartOfTest', 'EndOfTest', 'Log', 'StopProcessing', 'PauseProcessing' or 'ResumeProcessing'. Must subscribe to SSE topic to receive SSE messages. *AMPCS R8* valid values are 'SessionHeartbeat', 'TelemetryInputSummary', 'StartOfSession', 'EndOfSession' or 'Log'. SSE messages are included.
event_time	The time AMPCS generated the message (wall clock time). String in the format YYYY-DOYTHH:MM:SS.FFF.
summary	Message summary that will vary according to the message type. String.

9.13 Virtual Datatable Definition – Packet

Column Name	Description
record_type	Value is always the literal string 'packet'.
session_id	The AMPCS session ID. Positive integer.
session_host	The name of the host on which the session was run. String, maximum of 63 characters.
rct	The AMPCS Record Creation Time of the packet record. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings.
scet	The Spacecraft Event Time of the packet. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings.
lst	The Local Solar Time of the packet. String in the format SOL-XXXXMHH:mm:ss.SSS.
ert	The Earth Receive Time for the packet. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings.
sclk	The Spacecraft Clock Time of the packet. String of the format CCCCCCCCCC-FFF[FF] or CCCCCCCCCC.FFF[?FF], depending on project settings.
vcid	Virtual Channel ID on which the packet was received. Unsigned integer.
dss_id	Receiving station id. Unsigned integer.
apid	Application Process ID of the packet. Unsigned integer.
apid_name	Telemetry dictionary name that maps to the packet Application Process ID. String, maximum of 64 characters.
from_sse	Flag indicating that the packet is from the Simulation Support System (SSE) as opposed to the flight system. Value can be 'true' or 'false'.
spsc	Source packet sequence counter. Unsigned integer.
length	Packet data length, in bytes. Includes the CCSDS header. If the —restoreBodies option is supplied, this length will be adjusted to account for any packet headers or trailers re-attached to the packet body upon query. Unsigned integer.
source_vcfc	Virtual channel frame counter of the telemetry frame containing the packet. Unsigned integer.

9.14 Virtual Datatable Definition – PacketSummaryEvent

Column Name	Description
record_type	Value is always the literal string 'packet_summary_event'.
session_id	The AMPCS session ID. Positive integer.
vcid	Virtual Channel ID on which the packet was received. Unsigned integer.
apid	Application Process ID of the packet. Unsigned integer.
apid_name	Telemetry dictionary name that maps to the packet Application Process ID. String, maximum of 64 characters.
spsc	Source packet sequence counter per APID. Unsigned integer.
last_ert_time	The Earth Receive Time of the latest packet received for the selected virtual channel and frame type. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings.
last_sclk_time	The Spacecraft Event Time of the latest packet received for the selected virtual channel and APID. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings.
last_scet_time	The Spacecraft Event Time of the latest packet received for the selected virtual channel and APID. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings.
last_lst_time	The Local Solar Time of the latest packet received for the selected virtual channel and APID. String in the format of SOL-XXXXMHH:mm:ss.SSS. Used only for lander missions in the 'surface' venue.
event_time	The time AMPCS generated the packet summary message (wall clock time). String in the format YYYY-DOYTHH:MM:SS.FFF.
num_valid_packets	Total number of valid packets. Unsigned integer.
num_invalid_packets	Total number of invalid packets. Unsigned integer.
num_fill_packets	Total number of fill (non-data) packets. Unsigned integer.
num_frame_repeats	The number of times there are two frames in a row with the same sequence number. Unsigned integer.
num_frame_gaps	The number of gaps in the frame sequence number on a single virtual channel (note this field captures the number of gaps, not the number of missing frames). Unsigned integer.
num_station_packets	Number of station packets received (e.g., DSN monitor data or NEN status data), Unsigned integer.
num_frame_regressions	The number of times the frame sequence number on a single virtual channel regresses. Unsigned integer.
from_sse	Flag indicating if the packet comes from SSE or FSW. Boolean value in which if set to 'true' means packet came from SSE.

instance_count	The number of packets for the APID associated with the AMPCS session. Unsigned integer.

9.15 Virtual Datatable Definition – Session

Column Name	Description
record_type	Value is always the literal string 'session'.
number	The AMPCS session ID. Positive integer greater than zero.
name	The session name. String, maximum of 64 characters.
type	The session type. String, maximum of 64 characters.
description	The session description. String, maximum of 255 characters.
user	User name under which the session was run. String, maximum of 32 characters.
host	Host on which the session was run. String, maximum of 64 characters.
downlink_connection_type	Downlink connection type to the telemetry source. String, maximum of 32 characters.
uplink_connection_type	Uplink connection type to the command sink.. String, maximum of 32 characters
start_time	Time at which the session started. UTC string in the format YYYY-MM-DDTHH:mm:ss or YYYY-DOYTHH:mm:ss, depending on project settings.
end_time	Time at which the session ended. UTC string in the format YYYY-MM-DDTHH:mm:ss or YYYY-DOYTHH:mm:ss, depending on project settings.
output_directory	Directory to which session output was written. String, maximum of 1024 characters.
output_dir_overridden	“true” if the user overrode the output directory, “false” otherwise.
fsw_version	Version of the flight software dictionary used by this session. String, maximum of 64 characters.
sse_version	Version of the simulation support equipment dictionary used by this session. String, maximum of 64 characters.
fsw_dictionary_directory	Root flight software dictionary directory for this session. String, maximum of 1024 characters.
sse_dictionary_directory	Root simulation support equipment directory for this session. String, maximum of 1024 characters.
input_format	The input format of the telemetry for this session. Value can be 'RAW_TF', 'RAW_PKT', SFDU_TF', 'SFDU_GIF', 'SFDU_PKT', 'MS_TF', 'CMD_ECHO','LEOT_TF’, or 'UNKNOWN'.
venue_type	The venue in which AMPCS was run in order to generate data. (Values are configurable but tend to be of the form “TESTBED”, “ATLO”, “OPS”, etc.)
testbed_name	Name of the mission testbed on which the session was run. String, maximum of 32 characters. Values are project-specific.
downlink_stream_id	Specific telemetry stream identifier. String, maximum of 16 characters.
spacecraft_id	Numeric spacecraft ID. Unsigned integer.

ampcs_version	Version of AMPCS used to process this session. String, maximum of 16 characters.
full_name	Full name of session. String, maximum of 255 characters.
fsw_downlink_host	FSW downlink telemetry source. String, maximum of 64 characters.
fsw_uplink_host	FSW uplink command sink. String, maximum of 64 characters.
fsw_downlink_port	FSW downlink port number on host to connect to. Unsigned integer.
fsw_uplink_port	FSW uplink port number on host to connect to. Unsigned integer.
sse_host	SSE telemetry source and command sink. String, maximum of 64 characters.
sse_downlink_port	SSE downlink port number on host to connect to. Unsigned integer.
sse_uplink_port	SSE uplink port number on host to connect to. Unsigned integer.
input_file	Full path to input file if used. String, maximum of 1024 characters.
topic	Session topic used with JMS messaging. String, maximum of 128 characters.
jms_subtopic	Session subtopic used with JMS messaging. String, maximum of 128 characters.
session_dss_id	Station ID. 0 is used when unspecified or not applicable. Non-negative integer.
session_vcid	Virtual channel ID. Non-negative integer. “” when unspecified or not applicable.
run_fsw_downlink	“true” if session incorporated a FSW downlink, “false” otherwise.
run_sse_downlink	“true” if session incorporated a SSE downlink, “false” otherwise.
run_uplink	“true” if session incorporated an uplink, “false” otherwise.
database_session_key	If the input type was DATABASE, the source session id, else “”. Integer greater than 0.
database_session_host	If the input type was DATABASE, the source host name, else “”. String, maximum of 64 characters.

9.16 Virtual Datatable Definition – ChannelLAD

Column Name	Description
record_type	Value is always the literal string 'eha'.
channel_id	The channel identifier from the telemetry dictionary. String, maximum 16 characters.
name	The long name of the channel from the telemetry dictionary. String, maximum 64 characters.
ert	The Earth Receive Time for the channel sample. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings.
scet	The Spacecraft Event Time of the channel sample. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings.
lst	The Local Solar Time of the channel sample, if available. String in the format SOL-XXXXMHH:mm:ss.SSS.
sclk	The Spacecraft Clock Time of the channel sample. String of the format CCCCCCCCCC-FFF[FF] or CCCCCCCCCC.FFF[?FF], depending on project settings.
eu_or_dn	The channel sample value, as an EU floating point value if available, otherwise as a Data Number. Boolean and Enumeration channel types are always returned as the DN value.
dn_alarm_state	Alarm status, for alarms computed on the Data Number. Value can be 'RED', 'YELLOW', or 'NONE'.
eu_alarm_state	Alarm status, for alarms computed on the Engineering Unit. Value can be 'RED', 'YELLOW', or “NONE”.

9.17 Virtual Datatable Definition – ChannelMinMax

Column Name	Description
record_type	Value is always the literal string 'eha'.
session_id	The AMPCS session ID. Positive integer greater than zero.
session_host	The name of the host on which the session was run. String, maximum 64 characters.
channel_id	The channel identifier from the telemetry dictionary. String, maximum 16 characters.
name	The long name of the channel from the telemetry dictionary. String, maximum 64 characters.
ert	The Earth Receive Time for the channel sample. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS[SSSS] or YYYY-DOYTHH:mm:ss.SSS[SSSS], depending on project settings. For TSDB queries, time is returned as epoch seconds. Unsigned integer.
scet	The Spacecraft Event Time of the channel sample. UTC string in the format YYYY-MM-DDTHH:mm:ss.SSS or YYYY-DOYTHH:mm:ss.SSS, depending on project settings. For TSDB queries, time is returned as epoch seconds. Unsigned integer.
lst	The Local Solar Time of the channel sample, if available. String in the format SOL-XXXXMHH:mm:ss.SSS.
sclk	The Spacecraft Clock Time of the channel sample. String of the format CCCCCCCCCC-FFF[FF] or CCCCCCCCCC.FFF[?FF], depending on project settings.
eu_or_dn	The channel sample value, as an EU floating point value if available, otherwise as a Data Number. Boolean and Enumeration channel types are always returned as the DN value.
dn	The channel sample Data Number. This is the raw value received from the spacecraft or SSE. Data type is channel dependent.
eu	For numeric channels, this is the floating point Engineering Unit value for the channel, if an EU is defined. Empty otherwise.
dn_alarm_state	Alarm status, for alarms computed on the Data Number. Value can be 'RED', 'YELLOW', or 'NONE'.
eu_alarm_state	Alarm status, for alarms computed on the Engineering Unit. Value can be 'RED', 'YELLOW', or “NONE”.

10 Appendix: Examples

Note that the example URLs here are not meant to be accessed and are only shown for structure.

10.1 Representative examples

```
GET
https://ciedevweb1.jpl.nasa.gov:8443/mcws/gov/nasa/jpl/mcs/project/msl/
testbed1/tlm/qdb/Channel?filter=(session_id=2306,type!=(STATUS,BOOLEAN)
,channel_id=(P-*))&select=(channel_id,ert,scet,dn,eu)&sort=(ert)

GET
https://ciedevweb1.jpl.nasa.gov:8443/mcws/gov/nasa/jpl/mcs/project/msl/
ops/tlm/lom/Channel?filter=(lst=SOL-
0514*,type!=(STATUS,BOOLEAN),channel_id=(P-
*))&select=(channel_id,ert,scet,dn,eu)&sort=(ert)
```

10.2 Life of mission queries

These are actual user-supplied queries, extracted from msl Elastic Search prototype effort and run against the msl life of mission exported data set, converted to the interface spec established in this document.

EHA Data within a particular value range at a particular instant of time on Mars, in csv format

```
GET
https://ciedevweb1.jpl.nasa.gov:8443/mcws/gov/nasa/jpl/mcs/project/msl/ops/tlm/lom/Channel?filter=(lst=SOL-%200626M23:59:51.362,dn%3C=724,dn%3E=632,channel_id=(THRM-2562,THRM-%202582))&select=(channel_id,lst,ert,scet,dn,eu)&output=csv
```

Same, in JSON format

```
GET
https://ciedevweb1.jpl.nasa.gov:8443/mcws/gov/nasa/jpl/mcs/project/msl/ops/tlm/lom/Channel?filter=(lst=SOL-%200626M23:59:51.362,dn%3C=724,dn%3E=632,channel_id=(THRM-2562,THRM-%202582))&select=(channel_id,lst,ert,scet,dn,eu)&output=json
```

One hour's worth of EHA data for a single channel, in csv format

```
GET
https://ciedevweb1.jpl.nasa.gov:8443/mcws/gov/nasa/jpl/mcs/project/msl/ops/tlm/lom/Channel?filter=(ert%3E=2013-219T00:00:00.000,ert%3C2013-
219T01:00:00.000,channel_id=(THRM-2627))&select=(channel_id,ert,scet,dn,eu)
```

Ten Mars hours' worth of EVR data from two specific sessions for two specific EVRs

```
GET
https//ciedevweb1.jpl.nasa.gov:8443/mcws/gov/nasa/jpl/mcs/project/msl/ops/tlm/lom/EventRecord?filter=(session_id=(6889,6890),lst%3E=SOL-0626M00:00:00.0,lst%3CSOL-0626M10:00:00.0,event_id=(790726143,1280031383)&select=(event_id,scet,message)&output=csv
```

One hour's worth of EVR data (all columns) for a category of EVRs, in JSON format

`GET`

https://ciedevweb1.jpl.nasa.gov:8443/mcws/gov/nasa/jpl/mcs/project/msl/ops/tlm/lom/EventRecord?filter=(ert%3E=2013-219T00:00:00.000,ert%3C2013-219T01:00:00.000,module=cbm)&output=json