

SEMI A4-0821

SPECIFICATION FOR THE AUTOMATED TEST EQUIPMENT TESTER EVENT MESSAGING FOR SEMICONDUCTORS (TEMS)

This Standard was technically approved by the Automated Test Equipment Global Technical Committee. This edition was approved for publication by the global Audits and Reviews Subcommittee on January 8, 2021. Available at www.semiviews.org and www.semi.org in August 2021; originally published April 2021.

NOTICE: The designation of SEMI A4 was updated during the 0821 publishing cycle to reflect the publication of SEMI A4.1.

NOTICE: Paragraphs entitled ‘NOTE:’ are not an official part of this Standard or Safety Guideline and are not intended to modify or supersede the official Standard or Safety Guideline. These have been supplied by the global technical committee to enhance the usage of the Standard or Safety Guideline.

Table of Contents

1 Purpose	3
2 Scope	3
3 Limitations	3
4 Referenced Standards and Documents	3
5 Terminology	3
6 Conventions	3
7 Architecture Overview.....	4
7.1 Test Cell Communication Overview	4
7.2 TCS Overview.....	4
7.3 TCS Software Overview	5
7.4 Message Overview.....	5
7.5 Message Transmission Process Overview.....	6
7.6 Message Transmission Process Inside TCS	6
7.7 Initialization Phase with a DAS	8
7.8 Queue Management.....	10
7.9 TCS States	11
7.10 TCS Sends Compression Data.....	13
7.11 Message ID.....	13
7.12 Optional Parameter Syntax	13
8 TEMS Post Message Formats.....	13
8.1 TCS Initialization Message	13
8.2 TCS Shutdown Message	16
8.3 Status Message	18
8.4 Test Cell Configuration Message	19
8.5 User Status System Message	24
8.6 Tester Operating System Message	25
8.7 Test Program Load Message.....	28

8.8 Lot Start Message.....	30
8.9 Run Start Message.....	34
8.10 Run End Message.....	38
8.11 Run Status Message.....	40
8.12 Test Start Message	43
8.13 Test End Message.....	45
8.14 Test Data Message	47
8.15 Halt Message.....	51
8.16 External Message	52
8.17 Peripheral Message.....	54
8.18 Maintenance Message	55
8.19 Maintenance Data	57
8.20 Generic Message.....	58
8.21 Log Message.....	59
8.22 File Message	61
9 Related Documents.....	63
Appendix 1 List of Countries and Code	64
Appendix 2 Time Zone Coding	69
Appendix 3 MAC Address Format.....	78
Appendix 4 GPS Address Format.....	79

1 Purpose

1.1 Semiconductor test operations involving automated test equipment (ATE) today are experiencing increasing use of data for real-time data analysis and real-time ATE input and control, to improve test yield, throughput, efficiency, and product quality. At the same time, test equipment and test operations around the world use a diverse range of data formats, specifications, and interface requirements that create significant customer service and application engineering costs for ATE vendors, OSAT companies, IDM test operations, software providers, and handler equipment. The goal of this Specification is to create a vendor neutral way to collect test cell data.

2 Scope

2.1 This Specification describes the data communication between the test cell and an external server.

2.2 This Specification covers test cell data and operating conditions collection.

2.3 This Specification indicates the minimum interface requirements along with methods for adding custom capabilities.

NOTICE: SEMI Standards and Safety Guidelines do not purport to address all safety issues associated with their use. It is the responsibility of the users of the Documents to establish appropriate safety and health practices, and determine the applicability of regulatory or other limitations prior to use.

3 Limitations

3.1 None.

4 Referenced Standards and Documents

4.1 SEMI Standards and Safety Guidelines

NOTE 1: The SEMI Automated Test Equipment Global Technical Committee is developing a Rich Interactive Test Database Specification intended to be used in conjunction with this Standard.

4.2 ISO Standards¹

ISO 8601 — Date and Time Format

NOTICE: Unless otherwise indicated, all documents cited shall be the latest published versions.

5 Terminology

5.1 Abbreviations and Acronyms

5.1.1 *DAS* — data application server

5.1.2 *SECS* — Semiconductor Equipment Communication Standard

5.1.3 *TCS* — tester control system

5.1.4 *TEMS* — Tester Event Messaging for Semiconductors

5.2 Definitions

5.2.1 *data application server (DAS)* — software acting at the TEMS message server.

5.2.2 *heartbeat* — periodic status message sent at a controlled rate.

5.2.3 *message broker* — wide area message distribution system,

5.2.4 *tester control system (TCS)* — TEMS software running on a tester.

6 Conventions

6.1 None.

¹ International Organization for Standardization, ISO Central Secretariat, BIBC II, Chemin de Blandonnet 8, CP 401, 1214 Vernier, Geneva, Switzerland; Telephone: +41.22.749.01.11, <http://www.iso.org>

7 Architecture Overview

7.1 Test Cell Communication Overview

7.1.1 This Specification provides a new method for status and information transfer. This overlays the existing test cell and factory infrastructure with minimal impact on the currently running data collection and control structures.

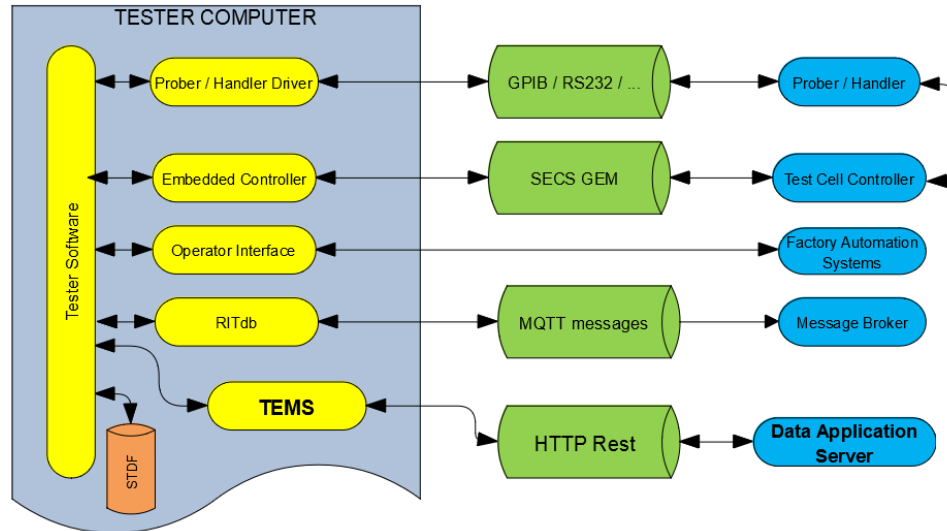


Figure 1
Overview of Test Cell Communication

7.1.2 This Specification provides the way the test cell system (TCS), (e.g., Tester), should communicate with a data application server (DAS).

7.1.3 TEMS software should not interfere with any current communication or automation system, for example, SECS GEM automation or prober communication.

7.2 TCS Overview

7.2.1 The basic topology of the system can include multiple TCS (test cells) and multiple DAS (servers). Each TCS can send data to multiple DAS and each DAS can receive data from multiple TCS.

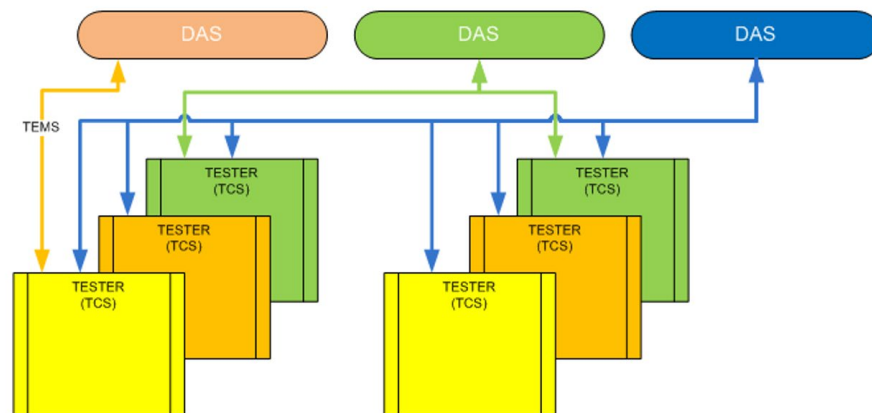


Figure 2
TEMS Topology

7.2.2 Each DAS is responsible for having enough bandwidth to support the required data flow.

7.2.3 The goal is to perform as many control actions as possible, as well as responses from the DAS to specific TCS messages for the following reasons:

7.2.3.1 For performance reasons we want to avoid web servers on the TCS.

7.2.3.2 The TCS will only accept control at specific times.

7.2.4 The DAS can be started at any time and stopped at any time.

7.2.5 When a DAS initiates communication with a tester, depending on the DAS recipe configuration file, the DAS will receive the previous messages queued.

7.2.6 The size of the queue can be different for each DAS defined.

7.3 TCS Software Overview

7.3.1 The TCS is in charge of:

1. Collecting data from the tester. This component is identified by the 'TCS Data Collector'.
2. Translating data collected into the appropriate TEMS format. This component is identified by the 'TCS Data Translator'.
3. Transmitting messages to the different DAS. This component is identified by the 'TCS DAS Push'.

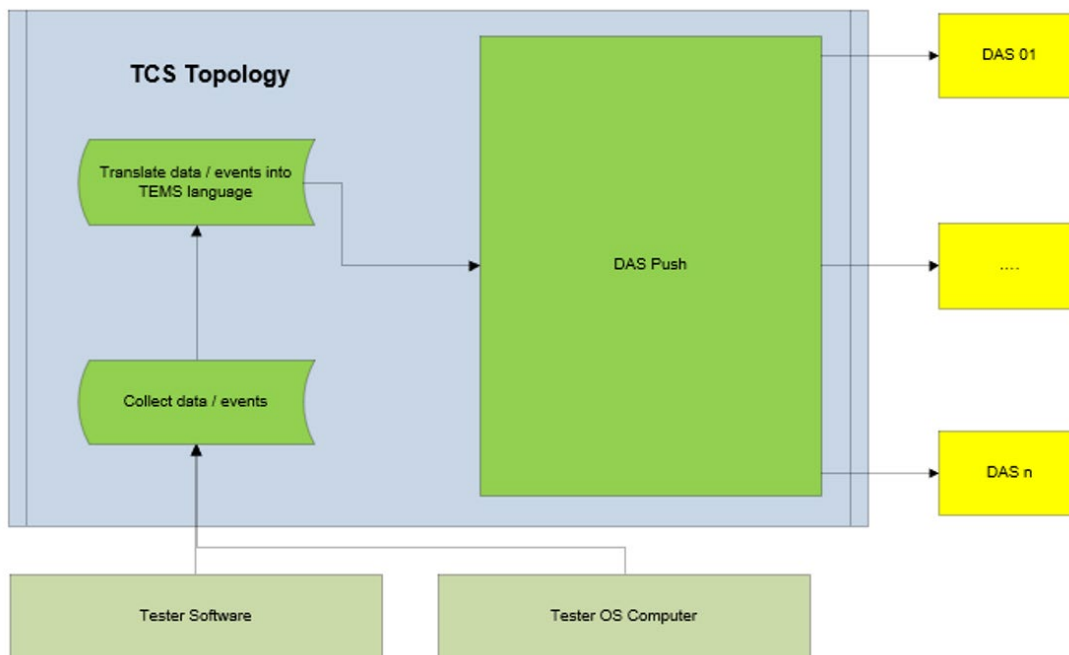


Figure 3
TCS Software Topology

7.3.2 The TCS should collect data with minimal impact on test cells.

7.4 Message Overview

7.4.1 All test cell communication messages are shown in Table 1 below. All messages other than Initialization, Shutdown, and Heartbeat are directly associated with Test Cell production environment state changes and test program status or are the result of data collection inquiries.

Table 1 Message List Description

<i>Message Name</i>	<i>Message Name Counterpart</i>	<i>Comment</i>
Initialization	Shutdown	System Startup/Shutdown
Status	-	Periodic Heartbeat
Test Cell Configuration	-	Tester cell configuration activity
Host Operating System	-	User Account Activity
Tester Operating System	-	Tester Software Activity
Test Program Load	-	Test Program Activity
Lot Start	Lot End	Lot activities
Run Start	Run End	Wafer/Strip activities
Run Status	-	Optional info to DAS during run
Test Start	Test End	Test a set of device activities
Test Data	-	Test Results
Halt	-	Any unexpected condition on the tester cell
External Source	-	Message from source external to the tester
Peripheral	-	Communication with a peripheral connected to the tester
Maintenance	Maintenance Data	Message related to the maintenance
Generic	-	For future/custom messages
Log	-	Logging messages
File	-	File activity related to the tester

7.5 Message Transmission Process Overview

7.5.1 Once a message is selected, it is sent to the associated DAS on any state-change, which includes the initial interest selection. For all messages, all available data is included with each message type based on the given TCS and running production environment capabilities.

7.5.2 Each DAS will return the list of messages that it wants to receive. Additionally, this list can be returned as part of any response to modify the list of messages that it wants to receive.

7.5.3 TCS performs POSTs at each Test Cell event on each interested DAS.

7.5.4 The TCS keeps some limited data persistence on the TCS for each DAS queue until the DAS acknowledges reception.

7.5.5 TCS only sends messages to the DAS, which has been subscribed.

7.5.6 The ‘TCS DAS Push’ is in charge of transmitting the messages to a DAS.

7.6 Message Transmission Process Inside TCS

7.6.1 The transmission of a message is done in two phases:

7.6.1.1 Internal transmission. The ‘TCS Das Push’ stores into a queue each message collected by the ‘TCS Data Collector’ before their transmission to the DAS to minimize impact.

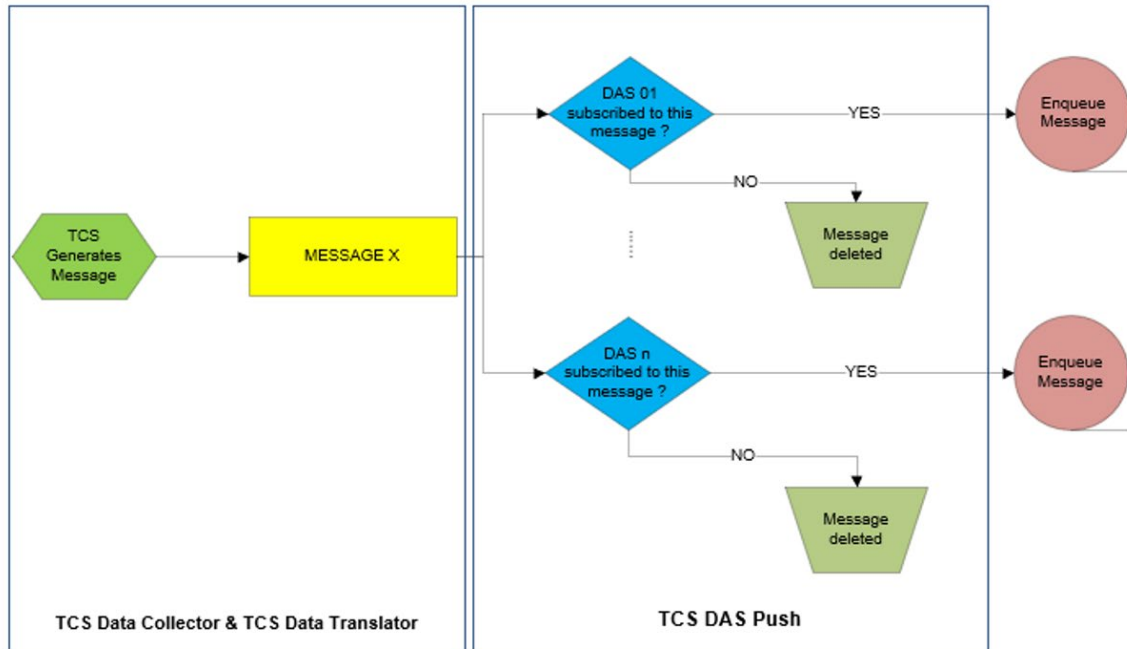


Figure 4
Message Transmission to TCS DAS Push

7.6.2 Transmission from TCS to DAS. The 'TCS DAS Push' oversees transmitting the data generated from the TCS to a DAS.

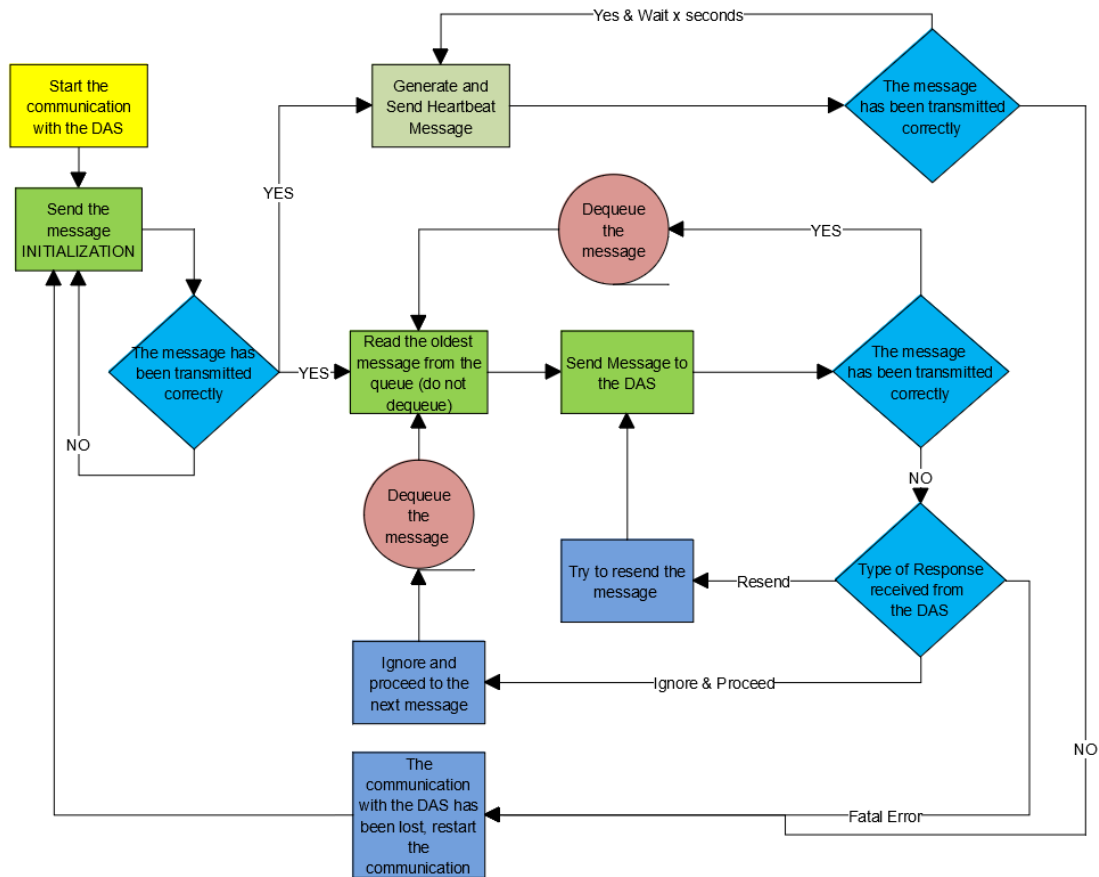


Figure 5
Message Transmission from TCS to DAS

7.7 Initialization Phase with a DAS

7.7.1 At initialization, the TCS knows the list of DAS it should connect to. The list of DAS is stored in a configuration file easily modified by the user. This file will also include the frequency of the heartbeat. This frequency will be used in case the DAS is not available for communicating. The TCS will try to reach the DAS after waiting a time defined by the TCS configuration. This file also defines the size of the queue for each DAS.

7.7.2 The TCS startup sequence always begins by sending an INITIALIZATION message to each DAS defined in the configuration file.

7.7.3 At the start, the TCS publishes the list of available messages in the INITIALIZATION message.

7.7.4 The DAS shall respond with the list of messages to which it subscribes.

7.7.5 The DAS cannot unsubscribe from the following messages: INITIALIZATION, STATUS, SHUTDOWN.

7.7.6 A heartbeat message provides the ability to re-establish interest in the event the DAS is not registered to any currently active Test Cell messages.

7.7.7 A heartbeat message provides the ability to disable any events the DAS has previously registered.

7.7.8 All messages are handled asynchronously by the TCS to minimize production throughput impact. Messages are processed with the standard buffering constructs applied. For a given DAS connection, all data transactions are serialized, but with respect to the current production flow, message handshakes can be separated by multiple device cycles.

7.7.9 As the arrival order is not guaranteed to match the execution order, the timestamp shall be used to determine the actual order.

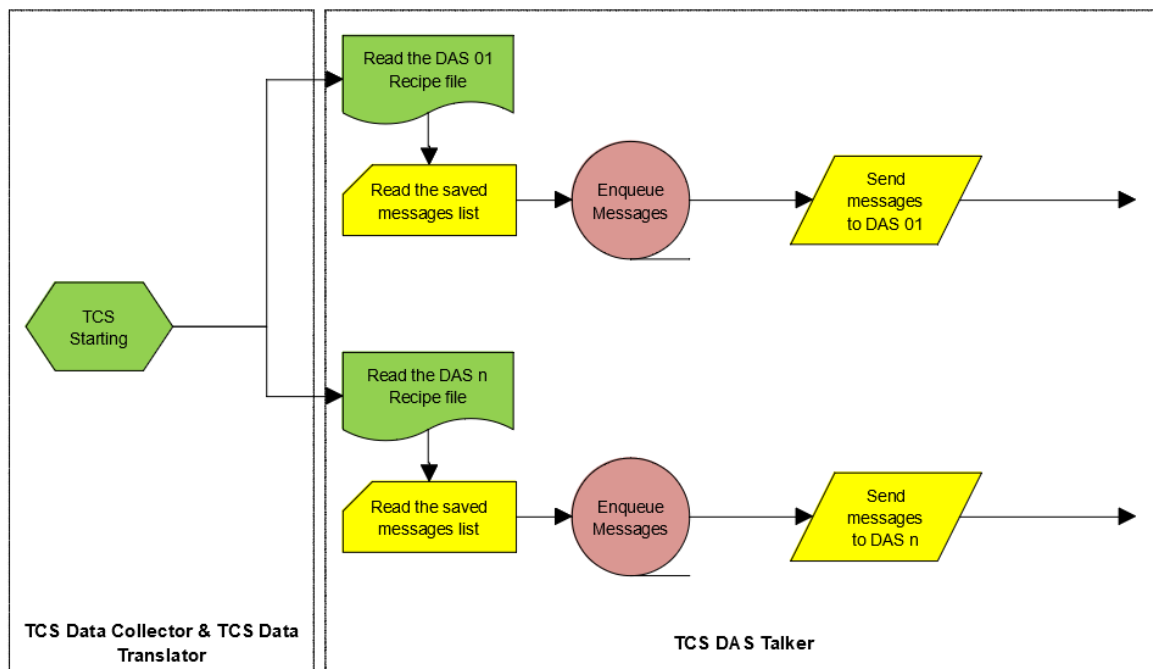


Figure 6
Initialization Phase

7.7.10 *DAS Response* — The DAS can respond in four different ways:

7.7.10.1 *DAS Accepts the Message*

7.7.10.1.1 DAS correctly receives the message from the TCS to a DAS.

7.7.10.1.2 Once the DAS receives the response, it removes the message from the queue.

7.7.10.2 *DAS Responds 'Resend'*

7.7.10.2.1 DAS requests to resend the message.

7.7.10.2.2 TCS tries to send the message again.

7.7.10.3 *DAS Responds 'Ignore & Proceed'*

7.7.10.3.1 DAS does not accept the message.

7.7.10.3.2 TCS removes this message from the queue.

7.7.10.3.3 TCS proceeds to the next message.

7.7.10.4 *DAS Responds 'Fatal Error'*

7.7.10.4.1 Communication with the DAS is interrupted.

7.7.10.4.2 TCS goes back to the initialization phase.

7.7.10.4.3 TCS tries to send the INITIALIZATION message.

7.7.11 *Shutdown Phase with DAS*

7.7.11.1 TCS Shutdown message ends the session when the TCS shuts down.

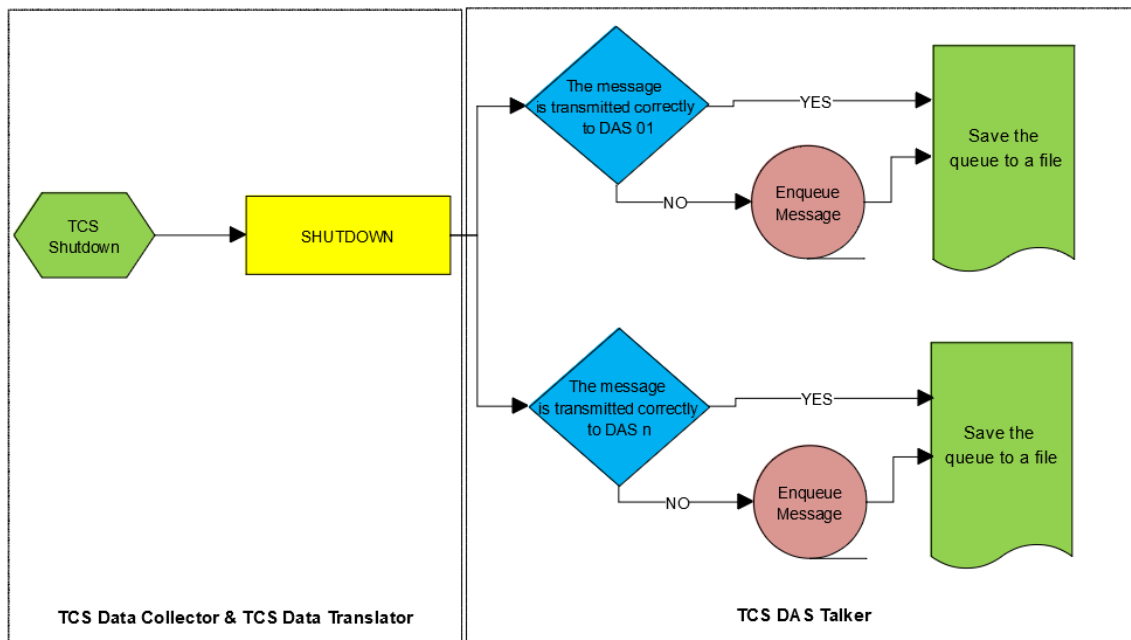


Figure 7
Shutdown Phase

7.7.12 The DAS can respond with three types of handshakes:

7.7.12.1 DAS accepts the message and TCS does not change the current state.

7.7.12.2 DAS does not accept the message and it is queued. TCS switches back to 'TEMS Startup' state.

7.7.12.3 DAS does not accept the message. TCS does not try to send it again and returns to the 'TEMS Startup' state.

7.7.13 The TCS should have a local log file that logs the message time, type, and destination. This will enable easier debug and tracing of issues.

7.7.14 The TCS keeps some limited data persistence on the TCS for each DAS queue until the DAS acknowledges reception.

7.8 Queue Management

7.8.1 For each DAS, the DAS Recipe configuration file defines the size of the queue associated with this DAS.

7.8.1.1 This size cannot be null. It should be at least equal to one.

7.8.1.2 The queue type is FIFO (first in, first out).

7.8.2 Each time TCS has to transmit a message, it should be added into the queue.

7.8.3 Each time TCS cannot transmit a message, it should be queued (depending on the handshake received from the DAS, Case 2).

7.8.4 If TCS is shutting down, the queue shall be saved persistently (e.g., in a file) to resend the contents of this queue after the INITIALIZATION message.

7.8.5 Once the queue is full, the oldest message should be removed from the queue. TCS should retain the number of lost messages. This number should be transmitted to the DAS once the new communication cycle is initiated into the INITIALIZATION message.

7.8.6 The heartbeat message (STATUS) should not be part of the queue and it does not change the number of lost messages.

7.8.7 TCS will flush the queue at any moment depending on the communication state with the DAS.

7.9 TCS States

7.9.1 TEMS defines up to six states.

7.9.1.1 When TCS software starts, TCS is in 'TEMS Startup' state. TCS stays in this phase up to the moment the TCS establishes communication with the DAS. TCS returns to this state if it loses communication with the DAS.

7.9.1.2 Once the communication is established, TCS switches to 'TCS Active' state. The system will stay in this state until an event occurs on the tester side.

7.9.1.3 TCS switches to 'TP Loaded' state once a test program has been loaded. TCS exits this state once the test program is unloaded.

7.9.1.4 During the testing phase TCS will go to 'RUNNING' state.

7.9.1.5 Any time the Maintenance tool is loaded or activated, TCS switches to 'MAINTENANCE' state.

7.9.1.6 When TCS is shutting down TCS switches to 'TCS INACTIVE' state.

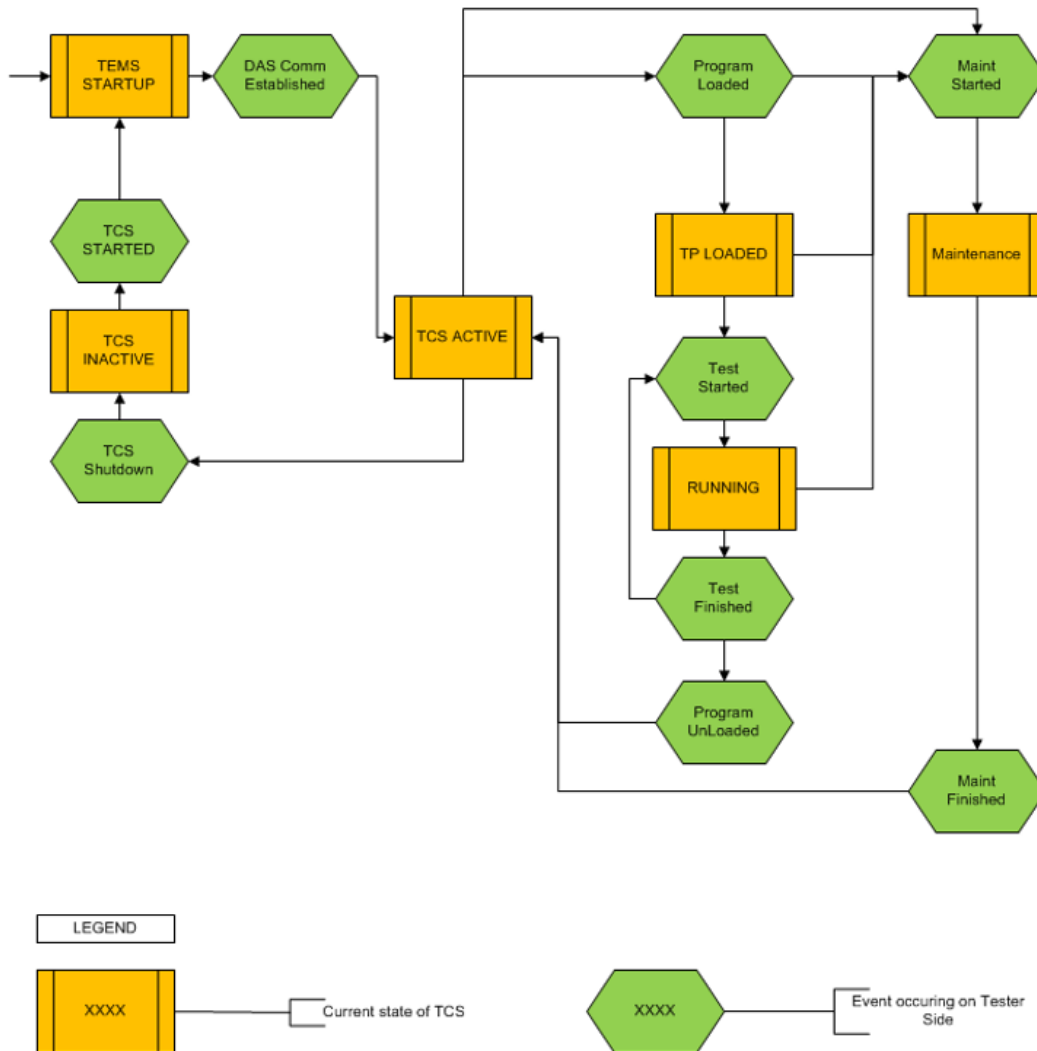


Figure 8
TCS State Machine

7.9.2 Some messages are allowed depending on the current state of the TCS.

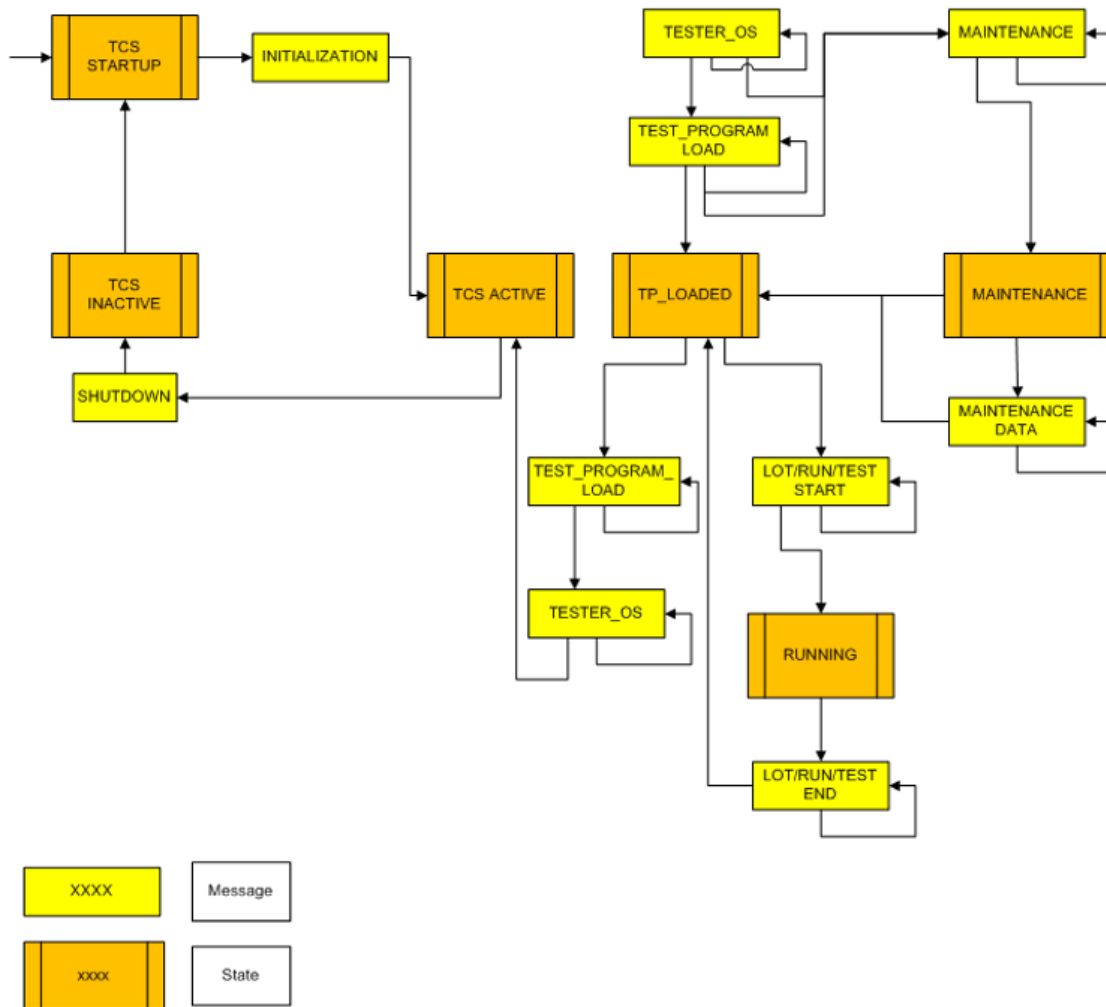


Figure 9
Messages Allowed During the Different States

7.9.2.1 INITIALIZATION shall be sent at 'TEMS Startup' state.

7.9.2.2 STATUS shall be sent at any state, except during 'TEMS STARTUP'.

7.9.2.3 STATUS is not mandatory during the 'RUNNING' state.

7.9.2.4 CONFIGURATION message should be sent after the INITIALIZATION Message.

7.9.2.5 CONFIGURATION shall be sent each time any item defined previously into CONFIGURATION message has been modified by the tester.

7.9.2.6 USER_STATUS shall be sent at any time after the INITIALIZATION message.

7.9.2.7 TESTER_OS can be sent only if TCS was in the 'TCS ACTIVE' or 'TP LOADED' states.

7.9.2.8 TEST_PROGRAM_LOAD can be sent only if TCS was in the 'TCS ACTIVE' or 'TP LOADED' states.

7.9.2.9 LOT_START, LOT_END, RUN_START, RUN_END, RUN_STATUS can be sent only if TCS was in the 'TP LOADED' or 'RUNNING' states.

7.9.2.10 TEST_START can be sent only from the 'TP LOADED', 'MAINTENANCE' and 'RUNNING' states.

7.9.2.11 TEST_END can be sent only from the 'RUNNING' state.

7.9.2.12 All other messages can be sent at any time except during the 'TEMS Startup' state.

7.10 TCS Sends Compression Data

7.10.1 Most of the messages between the TCS and DAS will be short; hence, not worthy of compression (i.e., the cost of compression in terms of CPU is not worth the resultant packet size). However, certain messages will benefit from compression and the manner in which compression can be handled needs to be standardized. It is not a requirement for a TCS or DAS to support compression, so the default for any communication is uncompressed.

7.10.2 Due to the fact that communication between TCS and DAS are via POST messages, the standard communication headers for compression availability cannot be used. To get around this problem, the TCS INITIALIZATION Message will be used to handshake on any possible compression that can be used between the two. When the TCS sends the INITIALIZATION message, it will include in the TCS_COMPRESSION_TYPES field the list of standard compression algorithms that it supports. In the response, the DAS will indicate in the COMPRESSION_TYPE field the single compression type that it will accept from the above list. If the DAS cannot handle any of the offered compression types from the TCS, then it will leave that field empty and no compression will take place.

7.10.3 After the INITIALIZATION message handshake is complete, the DAS should expect that any message following the INITIALIZATION can have a compressed body. The TCS will set the Content-Encoding header to the appropriate value agreed on during the initialization process whenever it wants to send compressed information.

7.11 Message ID

7.11.1 Message ID will contain an ID which is named TEST_CELL_ID.

7.11.1.1 Hostname can be used as TEST_CELL_ID.

7.11.2 Message ID can contain run type information as LOT_ID and/or RUN_ID.

7.12 Optional Parameter Syntax

7.12.1 None.

8 TEMS Post Message Formats

8.1 TCS Initialization Message

8.1.1 Description

8.1.1.1 This is the first message sent after the Controller powers up or the TCS is started. It is sent to all the DAS defined in the configuration file on the Controller. The payload contains the list of available message types supported by the TCS, and each DAS responds with the list of message types that it wants to receive. Simultaneously, the TCS broadcasts the list of control strings that the DAS can send to the TCS to control peripheral, such as 'Needle Clean' or 'Pause'.

8.1.1.2 After this initialization phase, the TCS is ready to send the appropriate messages to each DAS. At any time, the DAS can change the desired message list by using the response body of a message (see each message for details).

8.1.1.3 In case of a broken connection (if the TCS was not able to transmit the message) the TCS sends a new initialization message stating how many messages were lost.

8.1.2 Initiating Condition

8.1.2.1 In case of a broken connection (if the TCS was not able to transmit the message) the TCS sends a new initialization message stating how many messages were lost.

8.1.3 Post Syntax

8.1.3.1 Message ID

Table 2 ID for INITIALIZATION Message

Message_ID	/TEST_CELL/TEST_CELL_ID/INITIALIZATION
------------	--

8.1.3.2 Overview

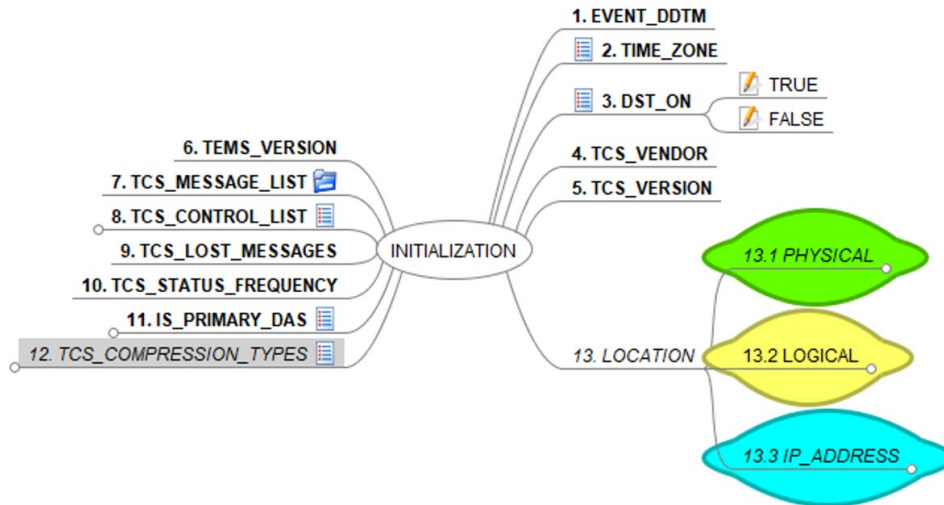


Figure 10
Initialization Message Overview

8.1.3.3 Message Body

Table 3 Body for INITIALIZATION Message

Order #	Name	Type	Description	Mandatory
1	EVENT_DDTM	DateTime(ms)	Event date and timestamp in ISO 8601 format when the initialization is complete. This timestamp is expressed in units including milliseconds by default but can be expressed in smaller units if desired.	Yes
2	TIME_ZONE	Enum	Local time zone where the tester is located. Refer to Appendix 2, Time Zone Coding.	Yes
3	DST_ON	Boolean	TRUE if Daylight Savings, otherwise FALSE.	Yes
4	TCS_VENDOR	String	Name of the company who provides the TCS software.	Yes
5	TCS_VERSION	String	Version of the current TCS software running.	Yes
6	TEMS_VERSION	String	The version of TEMS that the TCS is compliant with '0.94'.	Yes

Order #	Name	Type	Description	Mandatory
7	TCS_MESSAGE_LIST	Array[String]	Messages supported by this TCS • INITIALIZATION • SHUTDOWN • ...	Yes
8	TCS_CONTROL_LIST	Array[String]	Control strings supported by this TCS • NEEDLE_CLEAN • PAUSE • ...	Yes
9	TCS_LOST_MESSAGES	Integer	Number of lost messages in case of a broken communication [mandatory in case of recovery]. This is the number of messages that the TCS sent to the DAS when it lost communication with the DAS.	Yes
10	TCS_STATUS_FREQUENCY	Integer	Frequency of Status messages in seconds.	Yes
11	IS_PRIMARY_DAS	Boolean	Indicates if this DAS is the Primary DAS in terms of who can control the TCS, if any.	Yes
12	TCS_COMPRESSION_TYPES	Array[Enum]	Compression types supported by the TCS • GZIP • LZMA • DEFLATE • <custom>	No
13	LOCATION	Object	Where the tester is located.	No
13.1	PHYSICAL	Object	Define the physical location of the tester.	No
13.1.1	COUNTRY	Enum	Country with the standard ISO3. Refer to Appendix 1, List of Countries and Code.	No
13.1.2	STATE	String	State where the tester is located.	No
13.1.3	CITY	String	City where the tester is located.	No
13.1.4	ADDRESS	String	Postal address of the building.	No
13.1.5	GPS	Object	GPS location of the building.	No
13.1.5.1	LATITUDE	String	Latitude of the building in DD format. For further details, refer to Appendix 4.	No
13.1.5.2	LONGITUDE	String	Longitude of the building in DD format. For further details, refer to Appendix 4.	No
13.1.6	BUILDING	String	Building where the tester is located.	No
13.1.7	FLOOR	String	Floor where the tester is located.	No
13.2	LOGICAL	Object	Logical definition of the location of the tester.	Yes
13.2.1	FACILITY_ID	String	Unique company FACILITY_ID.	Yes
13.2.2	DOMAIN_ID	String	Shall be unique by FACILITY_ID.	Yes
13.2.3	CELL_ID	String	Shall be unique by DOMAIN_ID and FACILITY_ID (it can be the same as TEST_CELL_ID).	Yes
13.3	IP_ADDRESS	Array [Object]	List of IP addresses of network cards plugged into the tester.	Yes
13.3.1	ADDRESS	String	V4 or V6 format only.	Yes
13.3.2	MAC	String	Refer to Appendix 3.	Yes
13.3.3	SUBNET_MASK	String	Limits the number of valid IP's for this current network.	No

Order #	Name	Type	Description	Mandatory
13.3.4	GATEWAY	String	Address of the router to access to an IP address outside the MASK defined into SUBNET_MASK.	No
13.3.5	DESCRIPTION	String	Description of this network.	No
13.3.6	STATE	Enum	• CONNECTED • DISCONNECTED • DISABLED	No

8.1.4 Response

8.1.4.1 Response Syntax

Table 4 Body for INITIALIZATION Message Response

Order #	Name	Type	Description	Mandatory
1	DAS_VENDOR	String	Company that created the DAS.	Yes
2	DAS_NAME	String	Name of the current DAS.	Yes
3	DAS_VERSION	String	Version of the current DAS.	Yes
4	TEMS_VERSION	String	DAS should return the same version that TCS supports.	Yes
5	MESSAGE_LIST	Array[String]	List of messages desired by this DAS • Initialization • Shutdown • Etc.	Yes
6	COMPRESSION_TYPE	Enum	Compression types the DAS accepts. If there is no compression response, then the TCS does not compress packets. If there is a compression response but the TCS does not support compression, then the response is ignored. • gzip • lzma • deflate • <custom>	No

8.1.4.2 Response Behavior

8.1.4.2.1 If a transaction is successful, the TCS is ready to send all desired messages to this DAS.

8.1.4.2.2 If a transaction fails, the TCS retries based on the frequency and duration specified in the configuration file. It does not attempt to send any other message to this DAS until the Initialization message has been acknowledged successfully. If there is a mismatch, then the TCS disconnects and no longer sends messages to the DAS.

8.1.4.2.3 If it does not support the same version, then it should return the highest version it does support.

8.2 TCS Shutdown Message

8.2.1 Description

8.2.1.1 This is the last message sent before the Controller ends any session with a DAS. It happens for many reasons; for example, the Controller powers down or the TCS ends or wants to stop communicating with the DAS. The payload is the timestamp of the TCS when it sends the message and an optional REASON string.

8.2.2 Initiating Condition

8.2.2.1 TCS shutting down.

8.2.3 Post Syntax

8.2.3.1 Message ID

Table 5 ID for SHUTDOWN Message

Message_ID	<server>/TEST_CELL/<TEST_CELL_ID>/SHUTDOWN
------------	--

8.2.3.2 Overview



**Figure 11
SHUTDOWN Message Overview**

8.2.3.3 Message Body

Table 6 Body for SHUTDOWN Message

Order #	Name	Type	Description	Mandatory
1	EVENT_DTTM	DateTime(ms)	Event date and timestamp in ISO 8601 format when the shutdown command was sent. This timestamp is expressed in units including milliseconds by default but can be expressed in smaller units if desired.	Yes
2	REASON	String	Description indicating why the TCS is closing the connection.	No

8.2.4 Response

8.2.4.1 Response Syntax

Table 7 Body for SHUTDOWN Message Response

Order #	Name	Type	Description	Mandatory
---------	------	------	-------------	-----------

8.2.4.2 Response Behavior

8.2.4.2.1 Regardless of the result of a transaction, the TCS completes the shutdown.

8.2.4.2.2 The DAS does the necessary cleanup operations to avoid losing data. It is aware that this TCS is not transmitting.

8.3 Status Message

8.3.1 Description

8.3.1.1 This message provides a ‘heartbeat’ of the test cell to communicate the DAS that the test cell is still up even when it is idle and no other event is occurring. Additionally, like other messages, it allows each DAS to change the list of messages it subscribes to. Therefore, it provides a DAS that has subscribed to a few messages a way to change its subscription more often. To guarantee this feature, this message is automatically sent to all DAS that have completed the initialization handshake; this cannot be unsubscribed.

8.3.2 Initiating Conditions

8.3.2.1 Initiating conditions are sent regularly. Their frequency is defined in the setup file and not during testing.

8.3.3 Post Syntax

8.3.3.1 Message ID

Table 8 ID for STATUS Message

Message_ID	<server>/TEST_CELL/<TEST_CELL_ID>/STATUS
------------	--

8.3.3.2 Overview

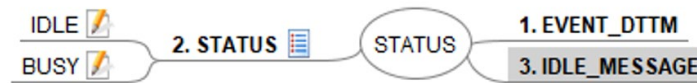


Figure 12
STATUS Message Overview

8.3.3.3 Message Body

Table 9 Body for STATUS Message

Order #	Name	Type	Description	Mandatory
1	EVENT_DTTM	DateTime(ms)	Event date and timestamp in ISO 8601 format. This timestamp is expressed in units including milliseconds by default but can be expressed in smaller units if desired.	Yes
2	STATUS	Enum	Status String • IDLE • BUSY IDLE: TCS and PC are online and available, but no tester OS, test program, or maintenance SW is running. BUSY: Tester OS is running with test program or maintenance.	Yes
3	IDLE_MESSAGE	String	When status is IDLE the string cannot be empty. When status is BUSY the string can be empty.	Yes

8.3.4 Response

8.3.4.1 Response Syntax

Table 10 Body for STATUS Message Response

Order #	Name	Type	Description	Mandatory
1	MESSAGE_LIST	Array[String]	List of messages desired by this DAS • INITIALIZATION • SHUTDOWN • Etc.	No

8.3.4.2 Response Behavior

8.3.4.2.1 If a transaction is successful, the TCS removes the relevant information from the queue for that DAS.

8.3.4.2.2 If the transaction fails, the TCS retries based on the frequency and size of the queue specified in the configuration file. When the maximum queue size is reached, the oldest data is deleted.

8.4 Test Cell Configuration Message

8.4.1 Description

8.4.1.1 This message is sent when the TCS detects a configuration change. The configuration includes any component of the complete test cell. If technically feasible, in addition to the tester HW configuration and licenses, it will include any peripheral external to the tester (such as handler or prober), any interface HW (such as load boards, probe cards, and sockets) and any SW information (such as Controller OS or driver revisions).

8.4.2 Initiating Conditions

8.4.2.1 At every Tester OS Session start.

8.4.2.2 When a configuration change is detected by the TCS (such as load board change if feasible).

8.4.3 Post Syntax

8.4.3.1 Message ID

Table 11 ID for CONFIGURATION Message

Message_ID	<server>/TEST_CELL/<TEST_CELL_ID>/CONFIGURATION
------------	---

8.4.3.2 Overview

8.4.3.2.1 The following schema shows an overview of the CONFIGURATION Message organization.

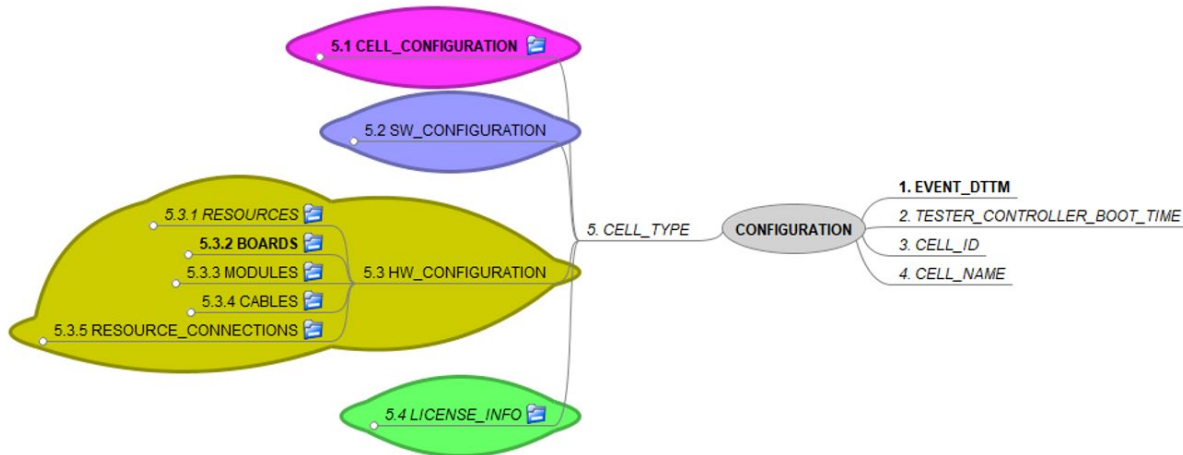


Figure 13
CONFIGURATION Message Overview

8.4.3.3 Message Body

Table 12 Body for CONFIGURATION Message

Order #	Name	Type	Description	Mandatory
1	EVENT_DTTM	DateTime(ms)	Event date and timestamp in ISO 8601 format. This timestamp is expressed in units including milliseconds by default but can be expressed in smaller units if desired.	Yes
2	TESTER_CONTROLLER_BOOT_TIME	DateTime(s)	Last boot time in ISO 8601 format. This timestamp is expressed in units including seconds by default but can be expressed in smaller units if desired.	No
3	CELL_ID	String	Unique ID to identify this test cell Customer provides this ID.	No
4	CELL_NAME	String	This data is provided by the Customer. By default the CELL_NAME of the tester is HOSTNAME.	No
5	CELL_TYPE	String	This data is provided directly by the Customer.	No
5.1	CELL_CONFIGURATION	Array[Object]	This configuration includes any component of the complete test cell.	Yes
5.1.1	ID	String	Unique ID for identifying this resource.	Yes
5.1.2	HOSTNAME	String	Hostname of this resource.	Yes
5.1.3	CLASS	Enum	• TESTER • HANDLER • DIB • HIB • CIB • <custom>	Yes

Order #	Name	Type	Description	Mandatory
5.1.4	PART_NUMBER	String	Part number of this resource.	No
5.1.5	INV_TYPE	String	Inventory Type, which defines the equipment connected. Provided by the vendor.	Yes
5.1.6	SERIAL_ID	String	Serial ID of this resource.	No
5.2	SW_CONFIGURATION	Object	List of software available.	No
5.2.1	SOFTWARE	Array[Object]	List of software available.	Yes
5.2.1.1	SOFTWARE_ID	String	Local Unique ID to identify this software.	Yes
5.2.1.2	NAME	String	Name describing this software.	Yes
5.2.1.3	CLASS	Enum	• TESTER_EXEC • LIBRARY • OS • DRIVER • OI • <custom>	Yes
5.2.1.4	VERSION	String	Any string identifying a unique version of this software.	No
5.2.1.5	BUILD	String	Build version.	No
5.2.1.6	LOCATION	String	Full path to software.	No
5.2.1.7	STATUS	Enum	• ENABLED • DISABLED	No
5.2.2	DELTAS	Array[Object]	List of Deltas available.	No
5.2.2.1	DELTA_ID	String	Unique ID to identify this software.	Yes
5.2.2.2	SOFTWARE_ID	String	Link to the software to which this patch is linked.	Yes
5.2.2.3	NAME	String	Name for this software.	Yes
5.2.2.4	VERSION		String that identifies a unique version of this delta software.	No
5.2.2.5	BUILD	String	Build ID	No
5.2.2.6	LOCATION	String	Full path to this software.	No
5.2.2.7	STATUS	Enum	• ENABLED • DISABLED	No
5.2.3	PATCHES	Array[Object]	List of patches available.	No
5.2.3.1	PATCH_ID	String	Unique ID to identify this software.	Yes
5.2.3.2	SOFTWARE_ID	String	Link to the software to which this patch is linked.	Yes
5.2.3.3	NAME	String	Name for this software patch.	Yes
5.2.3.4	VERSION		Any string which identifies a unique version of this patch software.	No
5.2.3.5	BUILD	String	Build ID	No
5.2.3.6	LOCATION	String	Full path to this software.	No
5.2.3.7	STATUS	Enum	• ENABLED • DISABLED	No
5.3	HW_CONFIGURATION	Object	Hardware configuration of the tester.	Yes
5.3.1	RESOURCES	Array[Object]	List of resources or instruments inside the tester.	No
5.3.1.1	RESOURCE_ID	String	Unique ID to identify this resource. This is defined by the vendor.	Yes

Order #	Name	Type	Description	Mandatory
5.3.1.2	RESOURCE_TYPE	String	Name for this resource.	Yes
5.3.1.3	PART_NUMBER	String	Part number of this resource.	No
5.3.1.4	SERIAL_NUMBER	String	Serial number of this resource.	No
5.3.1.5	CAL_STATUS	Enum	• PASS • FAIL	No
5.3.1.6	CAL_STATUS_DDTM_UTC	DateTime(s)	Time when calibration finished in ISO 8601 format. This timestamp is expressed in units including seconds by default but can be expressed in smaller units if desired.	No
5.3.1.7	VENDOR	String	Vendor of this resource.	No
5.3.2	BOARDS	Array[Object]	List of boards inside this tester.	No
5.3.2.1	BOARD_ID	String	Unique ID to identify this board. This is defined by the vendor. Create an algorithm to be sure this is a unique ID. Serial number in this case is not unique.	Yes
5.3.2.2	BOARD_TYPE	String	Name describing this board.	Yes
5.3.2.3	PART_NUMBER	String	Part number of this board.	No
5.3.2.4	SERIAL_NUMBER	String	Serial number of this board.	No
5.3.2.5	PCB_REV	String	PCB version used on this board.	No
5.3.2.6	FIRMWARE_VERSION	String	Firmware version.	No
5.3.2.7	CALIBRATIONS	Array[Object]	List of calibrations.	No
5.3.2.7.1	CAL_ID	String	Unique ID for this calibration Data.	Yes
5.3.2.7.2	CAL_TYPE	String	Type of calibration.	No
5.3.2.7.3	CAL_STATUS	Enum	• PASS • FAIL • UNKNOWN • INVALID • <custom>	No
5.3.2.7.4	CAL_REASON	String	Reason when calibration is not PASS.	No
5.3.2.7.5	CAL_STATUS_DTTM_UTC	DateTime(s)	Date when this board was calibrated in ISO 8601 format. This timestamp is expressed in units including seconds by default but can be expressed in smaller units if desired.	No
5.3.2.8	ASSEMBLY_REV	String	Assembly revision.	No
5.3.2.9	NUM_PINS	Integer	Number of Pins on this board.	No
5.3.2.10	VENDOR	String	Vendor name.	No
5.3.3	MODULES	Array[Object]	List of modules inside this tester.	No
5.3.3.1	MODULE_ID	String	Unique ID for defining this module.	Yes
5.3.3.2	MODULE_TYPE	String	Type of module.	No
5.3.3.3	PART_NUMBER	String	Part number of this module.	No
5.3.3.4	SERIAL_NUMBER	String	Serial Number of this module.	No
5.3.3.5	CAL_STATUS	Enum	• PASS • FAIL	No
5.3.3.6	CAL_STATUS_DDTM_UTC	DateTime(s)	Time when calibration is finished in ISO 8601 format. This timestamp is expressed in units including seconds by default but can be expressed in smaller units if desired.	No

Order #	Name	Type	Description	Mandatory
5.3.3.7	VENDOR	String	Vendor Name	No
5.3.4	CABLES	Array[Object]	Cables	No
5.3.4.1	CABLE_ID	String	Cable ID	Yes
5.3.4.2	CABLE_TYPE	String	Cable type	Yes
5.3.4.3	PART_NUMBER	String	Part Number of this cable.	No
5.3.4.5	SERIAL_ID	String	Serial ID of this cable.	No
5.3.5	RESOURCE_CONNECTION	Array[Object]	Resource Connection of this module.	No
5.3.5.1	CONNECTION_ID	String	Connection ID of this module.	No
5.3.5.2	RESOURCE_ID	String	Resource ID of this module.	No
5.3.5.3	SLOT_UNIQUE_ID	String	ID of this resource.	No
5.3.5.4	SLOT_NUMBER	String	Where is located this resource.	No
5.3.5.5	BOARD_ID	String	Board ID of this connection.	No
5.3.5.6	MODULE_ID	String	ID of this module.	No
5.3.5.7	CABLE_ID	String	ID of the Cable used.	No
5.4	LICENSE_INFO	Object	List of license available.	Yes
5.4.1	LICENSE_ID	String	License ID of this module.	Yes
5.4.2	LICENSE_NAME	String	License Name of this module.	Yes
5.4.3	DESCRIPTION	String	Description of this license.	No
5.4.4	VERSION	String	Version of this license.	No
5.4.5	LICENSE_EXP_TYPE	Enum	Type of expiration • PERM (Permanent) • TEMP (Temporary)	No
5.4.6	EXPIRATION_DATE	DateTime(d)	Expiration date of this license (if LICENSE_EXP_TYPE is TEMP). In ISO 8601 format. This timestamp is expressed in units including days by default but can be expressed in smaller units if desired.	No
5.4.7	LICENSE_TYPE	Enum	Type of license • NODE_LOCKED • FLOATING	No
5.4.8	STATUS	Enum	• ENABLED • DISABLED • EXPIRED	No
5.4.9	LICENSE_AVAILABLE	Integer	Number of license available.	No
5.4.10	LICENSE_USED	Integer	Number of license in used.	No
5.4.11	ORDER_ID	String	Order ID from where this license is part.	No

8.4.4 Response

8.4.4.1 Response Syntax

Table 13 Body for CONFIGURATION Message Response

Order #	Name	Type	Description	Mandatory
1	MESSAGE_LIST	Array[String]	List of messages desired by this DAS • INITIALIZATION • SHUTDOWN • Etc.	No

8.4.4.2 Response Behavior

8.4.4.2.1 If a transaction is successful, the TCS removes the relevant information from the queue for that DAS.

8.4.4.2.2 If the transaction fails, the TCS retries based on the frequency and size of the queue specified in the configuration file. When the maximum queue size is reached, the oldest data is deleted.

8.5 User Status System Message

8.5.1 Description

8.5.1.1 This message is sent when you log into or out of the tester computer.

8.5.2 Initiating Conditions

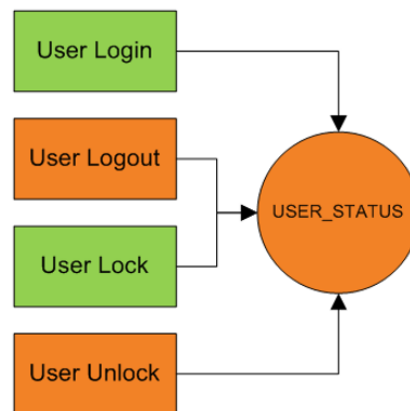


Figure 14
USER_STATUS Initiating Conditions

8.5.3 Post Syntax

8.5.3.1 Message ID

Table 14 ID for USER_STATUS Message

Message_ID	<server>/TEST_CELL/<TEST_CELL_ID>/USER_STATUS
------------	---