Test Specification

Senslify

By: Christen Ford

V 1.0.0

Introduction

This document contains the test suite and specification for the ‘Senslify’ Python web application. Test cases are written in a table in the following section and include a test identifier, feature identifier, input, expected output, and description. This document should be considered the de-facto standard with respect to guiding regression testing should future maintainers add additional features to the software over time. These test cases are extensive and are designed to test all features specified in the test plan for both correctness and coverage.

Note that many of these tests are written under the premise that the tester will be utilizing the layered architecture specified in the design document. For example, the DB-\*\*\* functions do not perform any actual validation on the inputs they receive. This is because the implementation is written in accordance with the architecture such that validation is performed on those inputs at a higher layer prior to these functions ever being called. Two facts arise from this: (1) performing input validation in the database functions will only slow the response time of the web server and (2) inputs are guaranteed to be correct so long as validation at the higher layers is correct.

This can be observed by noting that the implementation performs validation on the web browser side before sending requests via the WebSocket. Furthermore, additional validation is performed on requests as soon as they are received by the web server. Testing the database implementation using unit-level testing will return failures for many of these tests simply because validation is not performed at the database unit level due to the systems architecture. This should not be considered a failure on the implementations part but is in fact an intended feature.

Test Case Specification for ‘Senslify’ Python Web Application

Note: The CLI-\*\*\* features are meant to be manually tested while the web application is live inside the browser. This is why their input and output are both N/A.

Note: Each time you perform these tests, be certain you start with a new instance of the Senslify database.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Senslify Test Case Specification** | | | | |
| Test ID | Feature ID | Input | Expected Output | Description |
|  | CLI-000 | The tester leaves the sensor information page. | The WebSocket is no longer present in the rooms dictionary. | Tests that the WebSocket is torn down when the user leaves the sensor information page. |
|  | CLI-000 | The tester exits their web browser. | The WebSocket is no longer present in the rooms dictionary. | Tests that the WebSocket is torn down when the user exits the browser. |
|  | CLI-000 | The tester stops the web application. | The WebSocket is no longer present in the rooms dictionary. | Tests that the WebSocket is torn down when the web server goes down. |
|  | CLI-001 | The tester visits the sensor information page for groupid: 0 and sensorid: 0. | The rooms dictionary contains the testers WebSocket in the room with key (0, 0). | Tests that the user is subscribed to sensor updates when they visit the sensor information page. |
|  | CLI-001 | The tester manually visits the sensor information page URL with groupid: 0 and sensorid: -1 in the query parameters of the URL. | The tester receives an error page in response in their web browser. | Tests that the user is not subscribed to updates if they specify a non-existent sensor identifier. |
|  | CLI-001 | The tester manually visits the sensor information page URL with groupid: -1 and sensorid: 0 in the query parameters of the URL. | The tester receives an error page in response in their web browser. | Tests that the user is not subscribed to updates if they specify a non-existent group identifier. |
|  | CLI-002 | Tester supplies {“cmd”: “RESP\_ERROR”, “error”: “ERROR: This is a test message for TEST-0006!”} as input to the errorHandler function in the JS console in their web browser. | The string “ERROR: This is a test message for TEST-0006!” appears as an error in the error display at the top of the sensor information window. | Tests that the handler displays an error on the sensor information page if the tester supplies the correct command and an error message. |
|  | CLI-002 | Tester supplies {“cmd”: “TEST-0007”, “error”: “This is a test message for TEST-0007!”} to the errorHandler function in the JS console in their web browser. | No error is displayed at the top of the sensor information window. | Tests that the handler does not show an error on the sensor information page if the tester does not supply the correct command in a RESP\_ERROR command. |
|  | CLI-002 | Tester supplies {“cmd”: “TEST-0007”, “error”: “This is a test message for TEST-0007!”} to the errorHandler function in the JS console in their web browser. | No error is displayed at the top of the sensor information window. | Tests that the handler does not display an error if the tester does not supply an error message in a RESP\_ERROR command. |
|  | CLI-003 | Tester supplies {“cmd”: “RESP\_JOIN”, “join\_result”: false} to the joinHandler function in the JS console in their web browser. | N/A | Tests that the handler retries the join command if the web server returns false in a RESP\_JOIN command. |
|  | CLI-004 | Tester supplies {“cmd”: “RESP\_READING”, “sensorid”: 0, “groupid”: 0, “rtypeid”: 0, “val”: 50, “ts”: luxon.DateTime.now().toMillis().toUTC().toLocaleString(luxon.DateTime.TIME\_WITH\_SECONDS)} to the readingHandler function in the JS console in their web browser. | The reading is charted on the live chart as well as pushed to the top of the live reading list. | Tests that the handler updates the live chart and readings list when a correctly formatted reading is received in a RESP\_READING command. |
|  | CLI-005 | Tester uses the downloads modal to download readings uploaded prior to running this test. The tester must upload readings in the proper format to conduct this test for groupid: 0, sensorid: 0, rtypeid: 0. | The sensor readings uploaded prior to running this test are downloaded as a JSON file. | Tests that the handler downloads readings to a JSON file when receiving JSON data from the web server in a RESP\_DOWNLOAD command. |
|  | CLI-005 | Tester supplies {“cmd”: “RESP\_DOWNLOAD\_ERROR”, “error”: “This is a test message for TEST-0012!} to the onWSReceive function using the JS console in their web browser. | The string “This is a test message for TEST-0012!” appears as an error message in the downloads modal. | Tests that the handler displays an error message when the web server returns a RESP\_ERROR\_DOWNLOAD command. |
|  | CLI-006 | Tester uses the statistics modal to request statistics on sensor readings uploaded prior to running this test. The tester must upload readings in the proper format to conduct this test. | The statistics for sensor readings uploaded prior to this running this test are displayed on the statistics modal. | Tests that the handler updates the statistics display on the statistics modal box when receiving correctly formatted statistics information in a RESP\_STATS command. |
|  | CLI-007 | Tester uses the stream dropdown box to change the subscribed stream from “Temperature” to “Humidity”. This requires the tester to upload readings for sensorid: 0, groupid: 0, rtypeid: 0 and be on the sensor listing page for sensorid: 0, groupid: 0. | The live chart and sensor reading list are emptied of any plotted data. | Tests that the handler clears the live chart and reading list and adds received readings to the reading list when receiving readings in a RESP\_STREAM command. |
|  | CLI-008 | The tester sends a {“cmd”: “RQST\_CLOSE”} request to the web server using the JS console in their web browser then closes the WebSocket. | The live chart and sensor reading list are reset to their initial state (cleared of all held resources). | Tests that the live chart and reading list have their resources released when the WebSocket connection between the client and server are closed. |
|  | CLI-009 | The tester opens the sensor information page from the sensor listings page for sensorid: 0 and groupid: 0. This requires the tester provision a new group and sensor for groupid: 0 and sensorid: 0. | The live chart is animated and scrolling in real-time. | Tests that a RQST\_JOIN command is sent to the server and a RESP\_JOIN command is received in response from the server when the WebSocket connection between the client and server is opened. |
|  | CLI-010 | The tester sends a {“cmd”: “RESP\_JOIN”} command to the joinHandler using the JS console in their web browser. | N/A | Tests that a RESP\_JOIN coded message is sent to the joinHandler function when received by the client WebSocket. |
|  | CLI-010 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0}. | The reading is received by the client WebSocket and is shown in the live sensor reading list. (Due to the timestamp, it will not show on the live chart). | Tests that a RESP\_READING coded message is sent to the readingHandler function when received by the client WebSocket. |
|  | CLI-010 | The tester uses the reading type selector to switch the reading type to Humidity. | The live chart and sensor reading list are reset to contain the 100 most recent readings associated with the Humidity reading type. | Tests that a RESP\_STREAM coded message is sent to the streamHandler function when received by the client WebSocket. |
|  | CLI-010 | The tester uses the statistics modal to request statistics for the date range: 07/01/2021 – 07/02/2021. | The average, minimum, and maximum sensor readings are displayed on the statistics modal for the date range: 07/01/2021 – 07/02/2021. | Tests that a RESP\_SENSOR\_STATS coded message is sent to the statsHandler function when received by the WebSocket. |
|  | CLI-010 | The tester uses the download modal to request sensor readings for the date range: 07/01/2021 – 07/02/2021. | The tester is prompted to save the JSON file containing the readings for the date range: 07/01/2021 – 07/02/2021. | Tests that a RESP\_DOWNLOAD coded message is sent to the downloadHandler function when received by the WebSocket. |
|  | CLI-011 | The tester visits the sensor information page for sensorid: 0, groupid: 0. | The live sensor reading chart is displayed on the sensor information page. | Tests that a real-time ChartJS line chart is created on the UI when the createChart function is called. |
|  | CLI-013 | The tester sends the following commands to the web servers REST endpoint:  {“cmd”: “REST\_UPLOAD”, “sensorid”: 0, “rtypeid”: 0, “val”: 25, “ts”: 0},  {“cmd”: “REST\_UPLOAD”, “sensorid”: 0, “rtypeid”: 0, “val”: 75, “ts”: 5}. | “Most recent reading is above tolerated amount!” should be displayed as an alert at the top of the sensor information page. | Tests that an alert is displayed on the UI and added to the alert modal when the readingHandler function determines a reading should determine an alert. |
|  | CLI-014 | The tester uses the reading type selector to switch the reading type to Humidity. | The live chart is destroyed and redrawn while the sensor reading list is reset to contain the 100 most recent readings associated with the Humidity reading type. | Tests that the current line chart displayed on the UI is destroyed and replaced with a new chart when the reading type is switched using the appropriate UI control. |
|  | CLI-015 | The tester uses the reading type selector to switch the reading type to Humidity. | The live chart is destroyed and redrawn while the sensor reading list is reset to contain the 100 most recent readings associated with the Humidity reading type. | Tests that the current reading list displayed on the UI is cleared and loaded with sensor data corresponding to the selected reading type when the reading type is switched using the appropriate UI control. |
|  | CLI-016 | The tester submits the following commands to the web servers REST endpoint:  {“cmd”: “REST\_UPLOAD”, “sensorid”: 0, “rtypeid”: 0, “val”: 25, “ts”: 0},  {“cmd”: “REST\_UPLOAD”, “sensorid”: 0, “rtypeid”: 0, “val”: 75, “ts”: 5}.  The tester then opens the alerts modal and clicks on the “Clear Alerts” button. | The alerts modal is reset so it contains not alerts. | Tests that all alerts are cleared from the alerts modal when the user clicks the appropriate UI control in the alerts modal. |
|  | CLI-017 | The tester uses the download modal to request sensor readings for the date range: 07/01/2021 – 07/02/2021. | The tester is prompted to save the JSON file containing the readings for the date range: 07/01/2021 – 07/02/2021. | Tests that a RQST\_DOWNLOAD request is generated and sent to the web server when the user clicks the appropriate UI control in the downloads modal. |
|  | CLI-018 | The tester uses the statistics modal to request statistics for the date range: 07/01/2021 – 07/02/2021. | The average, minimum, and maximum sensor readings are displayed on the statistics modal for the date range: 07/01/2021 – 07/02/2021. | Tests that a RQST\_STATS request is generated and sent to the web server when the user clicks the appropriate UI control in the statistics modal. |
|  | CLI-019 | The tester uses the reading type selector to change the stream to Humidity. | The live chart and reading list are reset. The reading list is loaded with the top 100 most recent sensor readings for the Humidity reading type. | Tests that a RQST\_STREAM request is generated and sent to the web server when the user clicks the appropriate UI control on the sensor information page. |
|  | DB-001 | The tester creates an instance of the MongoProvider subclass with default parameters. | A MongoProvider instance | Tests that a subclass of the DatabaseProvider class is returned when the \_\_init\_\_ function is called. |
|  | DB-002 | The tester calls the db.MongoProvider.get\_connection function. | A MongoProvider instance | Tests that a context manager aware instance of a subclass of the DatabaseProvider class is returned when the get\_connection function is called. |
|  | DB-003 | The tester calls the close function when the database connection is open. | The database connection is closed. | Tests that the database connection is closed when the database connection is open and the close function is called. |
|  | DB-003 | The tester calls the close function when the database connection is closed. | DBError | Tests that a DBError is raised when the database connection is closed and the close function is called. |
|  | DB-004 | The tester calls calls the init function when the database connection is open. | The database contains the schema specified in the system design specification. | Tests that the database is initialized properly after calling the init function. |
|  | DB-005 | The tester supplies the following information to the function: {groupid: 0}. The following group must exist in the database before this test can be run:  {groupid: 0} | N/A | Tests that the group and corresponding sensors and sensor readings are deleted when valid input is supplied to the delete\_group function. |
|  | DB-006 | The tester supplies the following information to the function: {sensorid: 0, groupid: 0, rtypeid: 0, ts: 0}. The following sensor reading must exist in the database before this test can be run:  { sensorid: 0, groupid: 0, rtypeid: 0, ts: 0, val: 50} | The sensor reading corresponding to sensorid 0, groupid 0, rtypeid 0, and ts 0 are no longer in the database. | Tests that a reading is deleted if valid input is supplied to the delete\_reading function. |
|  | DB-007 | The tester supplies the following information to the function: { sensorid: 0, groupid: 0, rtypeid: 0}. The following sensor readings must exist in the database before this test can be run:  {sensorid: 0, groupid: 0, rtypeid: 0, ts: 0, val: 50},  { sensorid: 0, groupid: 0, rtypeid: 0, ts: 10, val: 50},  { sensorid: 0, groupid: 0, rtypeid: 0, ts: 20, val: 50} | All sensor readings corresponding to sensorid 0, groupid 0, and rtypeid 0 are no longer in the database. The integer value 3 is returned from the function. | Tests that the delete\_readings function deletes all correctly specified readings from the database. |
|  | DB-008 | The tester supplies the following information to the function: {rtypeid: 1}. The init DatabaseProvider function must be called before this test can be run. | The reading type corresponding to rtypeid 1 is no longer in the database. The integer value 1 is returned from the function. | Tests that the delete\_rtype function deletes a reading type and associated sensor readings from the database when valid input is supplied to this function. |
|  | DB-009 | The tester supplies the following information to the function: {groupid: 0, sensorid: 0}. The following group must exist in the database before this test can be run:  {groupid: 0} Additionally, the following sensor must exist in the database before this test can be run:  {groupid: 0, sensorid: 0}. | The sensor corresponding to groupid 0, sensorid 0 is no longer in the database. The integer value 1 is returned from the function. | Tests that the delete\_sensor function deletes a sensor and associated readings from the database when valid input is supplied to this function. |
|  | DB-010 | The tester supplies the following information to the function: {groupid: 0}. The following group must exist in the database before this test can be run:  {groupid: 0}. | True | Tests that the does\_group\_exist function returns True if a valid group identifier is supplied to this function. |
|  | DB-010 | The tester supplies the following information to the function: {groupid: 1}. The following group must not exist in the database before this test can be run:  {groupid: 1}. | False | Tests that the does\_group\_exist function returns False if an invalid group identifier is supplied to this function. |
|  | DB-011 | The tester supplies the following information to the function: {rtypeid: 0}. The The init DatabaseProvider function must be called before this test can be run. | True | Tests that the does\_rtype\_exist function returns True if a valid reading type identifier is supplied to this function. |
|  | DB-011 | The tester supplies the following information to the function: {rtypeid: 6}. | False | Tests that the does\_rtype\_exist function returns False if an invalid reading type identifier is supplied to this function. |
|  | DB-012 | The tester supplies the following information to the function: {groupid: 0, sensorid: 0}. The following group must exist in the database before this test can be run:  {groupid: 0}  Additionally, the following sensor must exist in the database before this test can be run:  {groupid: 0, sensorid: 0}. | True | Tests that the does\_sensor\_exist function returns True if valid input is supplied to this function. |
|  | DB-012 | The tester supplies the following information to the function: {groupid: 0, sensorid: 1}. The following group must not exist in the database before this test can be run:  {groupid: 0}  Additionally, the following sensor must exist in the database before this test can be run:  {groupid: 0, sensorid: 1}. | False | Tests that the does\_sensor\_exist function returns False if an invalid group identifier is supplied to this function. |
|  | DB-012 | The tester supplies the following information to the function: {groupid: 0, sensorid: 1}. The following group must exist in the database before this test can be run:  {groupid: 0}  Additionally, the following sensor must not exist in the database before this test can be run:  {groupid: 0, sensorid: 1}. | False | Tests that the does\_sensor\_exist function returns False if an invalid sensor identifier is supplied to this function. |
|  | DB-013 | The tester calls the function. The following group must exist in the database before this test can be run:  {groupid: 0}  No other group must exist in the database. | The function returns the integer value 0. | Tests that the find\_max\_groupid function returns the maximum group identifier stored in the database. |
|  | DB-014 | The tester supplies the following information to the function: {groupid: 0}. The following group must exist in the database:  {groupid: 0}  Additionally, the following sensor must exist in the database:  {groupid: 0, sensorid: 0}  No other sensors must exist in the database. | The function returns the integer value 0. | Tests that the find\_max\_sensorid\_in\_group function returns the maximum sensor identifier associated with the user-supplied group identifier stored in the database. |
|  | DB-015 | The tester calls the function. The following groups must exist in the database:  {groupid: 0, alias: “abc”},  {groupid: 1, alias: “bac”},  {groupid: 2, alias: “cba”} | [{groupid: 0, alias: “abc”},  {groupid: 1, alias: “bac”},  {groupid: 2, alias: “cba”}] | Tests that the get\_groups function returns all group identifiers and aliases stored in the database. |
|  | DB-016 | The tester calls the function. The DatabaseProvider init function must be called before this test can be run. | [{rtypeid: 0, rtype: “Temperature”},{rtypeid: 1, rtype: “Humidity”},{rtypeid: 2, rtype: “Visible Light”},{rtypeid: 3, rtype: “Infrared Light”},{rtypeid: 4, rtype: “Voltage”}] | Tests that the get\_rtypes function returns all reading type identifiers and aliases stored in the database. |
|  | DB-017 | The tester calls the function. The following sensors must exist before this test can be run:  {groupid: 0, sensorid: 0, alias: “abc”},  {groupid: 0, sensorid: 1, alias: “bac”} | [{groupid: 0, sensorid: 0, alias: “abc”},  {groupid: 0, sensorid: 1, alias: “bac”}] | Tests that the get\_sensors function returns all sensor identifiers and aliases associated with a user-supplied group identifier stored in the database. |
|  | DB-018 | The tester supplies the following information to the function: {sensorid: , groupid: , limit: 3}. The following group must exist before this function can be run: {groupid: 0, alias: “abc”} The following sensor must exist before this test can be run:  {groupid: 0, sensorid: 0, alias: “abc”}  The following sensor readings must exist before this test can be run:  {groupid: 0, sensorid: 0, rtypeid: 0, val: 0, ts: 0},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 10},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 20} | [{groupid: 0, sensorid: 0, rtypeid: 0, val: 0, ts: 0}, {groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 10}, {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 20}] | Tests that the function returns ‘limit’ number of sensor readings from the database for a user-supplied sensor identifier and group identifier. |
|  | DB-019 | The tester supplies the following information to the function: {groupid: 0, alias: “abc”} The preceding group must not exist before this test can be run. | The group { groupid: 0, alias: “abc”} is in the database. | Tests that the group is inserted when the valid parameters are supplied to this function. |
|  | DB-020 | The tester supplies the following information to the function: [{groupid: 0, sensorid: 0, rtypeid: 0, val: 0, ts: 0},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 10},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 20}] The following group must exist before this test can be run:  {groupid: 0, alias: “abc”}  The following sensor must exist before this test can be run:  {groupid: 0, sensorid: 0, alias: “bac”} | The sensor readings [{groupid: 0, sensorid: 0, rtypeid: 0, val: 0, ts: 0},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 10},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 20}] are in the database | Tests that the function inserts all readings passed to this function given valid input. |
|  | DB-021 | The tester supplies the following information to the function: {groupid: 0, sensorid: 0, alias: “bac”}  The following group must exist before this test can be run.:  {groupid: 0, alias: “abc”}  The sensor specified by this test case must not exist before this test can be run. | The sensor {groupid: 0, sensorid: 0, alias: “bac”} is in the database. | Tests that the function inserts a sensor given valid input. |
|  | DB-022 | The tester makes sure the database connection is open. The tester then calls the is\_open function. | True | Tests that the function returns True when the database connection is open. |
|  | DB-022 | The tester makes sure the database connection is closed. The tester then calls the is\_open function. | False | Tests that the function returns False when the database connection is not open. |
|  | DB-023 | The tester makes sure the database connection is closed. The tester then calls the open function. | N/A | Tests that the function opens the database connection when the database connection is not open. |
|  | DB-023 | The tester makes sure the database connection is open. The tester then calls the open function. | DBError | Tests that the function raises a DBError exception when the database connection is open. |
|  | DB-024 | The tester supplies the following information this the function: {}. The following group must exist in before this test can be run:  {groupid: 0, alias: “abc”}  The following sensors must exist before this test can be run:  {groupid: 0, sensorid: 0, alias: “bac”},  {groupid: 0, sensorid: 1, alias: “cba”}  The followings sensor readings must exist before this test can be run:  {groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 20},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 50},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 100, ts: 75},  {groupid: 0, sensorid: 1, rtypeid: 0, val: 10, ts: 20},  {groupid: 0, sensorid: 1, rtypeid: 0, val: 20, ts: 50},  {groupid: 0, sensorid: 1, rtypeid: 0, val: 30, ts: 75} | [{avg: 58.3333, max: 100, min: 25, groupid: 0, sensorid: 0}, {avg: 20, max: 30, min: 10, groupid: 0, sensorid: 1}] | Tests that the function returns the statistics for a group of sensors when the function is supplied valid input. |
|  | DB-025 | The tester supplies the following information this the function: {}. The following group must exist in before this test can be run:  {groupid: 0, alias: “abc”}  The following sensor must exist before this test can be run:  {groupid: 0, sensorid: 0, alias: “bac”}  The followings sensor readings must exist before this test can be run:  {groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 20},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 50},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 100, ts: 75} | {avg: 58.3333, max: 100, min: 25}. | Tests that the function returns the statistics for a sensor when valid input is supplied to this function. |
|  | DB-026 | The tester supplies the following information this the function: {sensorid: 0, groupid: 0, start\_ts: 0, end\_ts: 100}. The following group must exist in before this test can be run: {groupid: 0, alias: “abc”}  The following sensor must exist before this test can be run:  {groupid: 0, sensorid: 0, alias: “bac”}  The followings sensor readings must exist before this test can be run:  {groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 20},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 50},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 100, ts: 75} | [{groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 20},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 50},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 100, ts: 75}] | Tests that sensor readings are returned from the function when valid input is supplied to this function |
|  | FIL-000 | The tester supplies the following information to the function:  {dt: datetime.now(), fmt: “medium”, locale: ”en”} | String | Tests that the function returns a formatted string when valid input is supplied to this function. |
|  | FIL-000 | The tester supplies the following information to the function:  {dt: None, locale: ”en”} | Exception | Tests that the function fails when an invalid timestamp is supplied to this function. |
|  | FIL-000 | The tester supplies the following information to the function:  {dt: datetime.now(), locale: “en-US”} | Exception | Tests that the function fails when an invalid locale is supplied to this function. |
|  | FIL-001 | The tester supplies the following information to the function:  {dt: datetime.now(), fmt:, locale:} | String | Tests that the function returns an `i18n` compliant datetime string when valid input is supplied to this function. |
|  | FIL-001 | The tester supplies the following information to the function:  {dt: None, fmt: “medium”, locale: “en”} | Exception | Tests that the function fails when an invalid datetime is supplied to this function. |
|  | FIL-001 | The tester supplies the following information to the function:  {dt: datetime.now(), fmt: “”, locale: “en”} | Exception | Tests that the function fails when an invalid format string is supplied to this function. |
|  | FIL-001 | The tester supplies the following information to the function:  {dt: datetime.now(), fmt: “medium”, locale: “en-USK”} | Exception | Tests that the function fails when an invalid locale is supplied to this function. |
|  | FIL-002 | The tester supplies the following information to the function:  {reading: {“groupid”: 0, “sensorsid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0}} | Exception | Tests that the function returns a formatted string representing the reading when a valid sensor reading is supplied to this function. |
|  | FIL-002 | The tester supplies the following information to the function:  {reading: None} | Exception | Tests that the function fails when an invalid sensor reading is supplied to this function. |
|  | IND-000 |  | String | Tests that the function returns a URL route for a sensor when valid parameters are supplied to this function. |
|  | IND-000 |  | Exception | Tests that the function fails when invalid group information is supplied to this function. |
|  | IND-000 |  | aiohttp Response object | Tests that the function fails when an invalid request is passed to this function. |
|  | IND-001 |  | aiohttp Response object | Tests that the function returns the index page in an aiohttp Response object when valid parameters are supplied to this function. |
|  | IND-001 |  | aiohttp Response object | Tests that the function returns an error message and status code inside the body of an aiohttp Response when an invalid request is passed to this function. |
|  | INI-000 | Tester supplies the following information to the function:  {conn\_str: “mongodb://127.0.0.1:27017”, db\_provider: “MONGO”, auth\_required: False} | A DatabaseProvider instance | Tests that a DatabaseProvider subclass can be successfully created when valid parameters are supplied to this function. |
|  | INI-000 | Tester supplies the following information to the function:  {conn\_str: “”, db\_provider: “MONGO”, auth\_required: False} | Exception | Tests that the function fails when a poorly formed connection string is supplied to this function. |
|  | INI-000 | Tester supplies the following information to the function:  {conn\_str: “mongodb://127.0.0.1:27017”, db\_provider: “REDIS”, auth\_required: False} | ValueError | Tests that the function fails when an invalid database provider is provided to this function. |
|  | INI-001 | The tester calls the function. | “192.168.0.xxx” | Tests that the function is able to retrieve the local IP address of the web server. The tester must know the IP address of the server. |
|  | INI-002 | The tester calls the function. | aiohttp Application object | Tests that the function can successfully build and return an aiohttp Application instance corresponding to the web application using the default configuration file. |
|  | INI-002 | The tester supplies the path to their own configuration to the function. | aiohttp Application object | Tests that the function is able to successfully build and return an aiohttp Application instance corresponding to the web application using a user-supplied configuration file. |
|  | INI-003 | The tester calls the function. | N/A | Tests that the web application can be started using the default configuration file. |
|  | INI-003 | The tester supplies the path to their own configuration to the function. | N/A | Tests that the web application can be started using a user-provided configuration file. |
|  | RST-000 | The tester calls the function. | String | Tests that the function returns a 3-gram alias if the user does not supply any value for n. |
|  | RST-000 | The tester supplies the following information to the function:  {n: 3}. | String | Tests that the function defaults to creating a 3-gram alias if n is invalid. |
|  | RST-001 | The tester supplies a valid request as well as the following information to the function:  {sensorid: 0, groupid: 0, start\_ts: 0, end\_ts: 100}. The following group must exist:  {groupid: 0}  The following sensor must exist:  {groupid: 0, sensorid: 0}  The following sensor readings must exist:  {groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 0},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 50}  {groupid: 0, sensorid: 0, rtypeid: 0, val: 100, ts: 100} | aiohttp Response object | Tests that the function returns an aiohttp Response body containing a serialized JSON array of sensor readings when valid parameters are supplied to this function. |
|  | RST-002 | The tester supplies a valid request as well as the following information to the function:  {target: “groups”} The following groups must exist:  {groupid: 0, alias: “abc”},  {groupid: 1, alias: “bac”},  {groupid: 2, alias: “cba”} | aiohttp Response object | Tests that the function returns an aiohttp Response object where the body contains a serialized JSON array of sensor readings when valid parameters are supplied to this function. |
|  | RST-003 | The tester supplies a valid request as well as the following information to the function:  {target: “sensor”, groupid: 0, sensorid: 0, rtypeid: 0, start\_ts: 0, end\_ts: 100}. The following group must exist:  {groupid: 0}  The following sensor must exist:  {groupid: 0, sensorid: 0}  The following sensor readings must exist:  {groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 0},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 50}  {groupid: 0, sensorid: 0, rtypeid: 0, val: 100, ts: 100} | aiohttp Response object | Tests that the function returns an aiohttp Response object where the body contains a serialized JSON object representing the statistics for the target sensor when valid parameters are supplied to this function. |
|  | RST-004 | The tester supplies a valid request as well as the following information to the function. {target: “group”}. No groups must exist in the system. | aiohttp Response object | Tests that the function returns an aiohttp Response object where the body contains a serialized JSON object containing the sensor identifier and group alias. |
|  | RST-005 | The tester supplies a valid request as well as the following information to the function:  {params: [{groupid: 0, sensorid: 0, rtypeid: 0, val: 25, ts: 0},  {groupid: 0, sensorid: 0, rtypeid: 0, val: 50, ts: 50}  {groupid: 0, sensorid: 0, rtypeid: 0, val: 100, ts: 100}]}. | aiohttp Response object | Tests that the function returns an aiohttp Response body containing a serialized JSON object containing an upload status code and message when valid parameters are supplied to this function. |
|  | RST-006 | The tester supplies a valid request to the function. | aiohttp Response object | Tests that the function returns the results of a handler when the request body contains a valid parameter mapping. |
|  | SEN-000 | Tester supplies a valid request to the function by visiting the sensor information page through their web browser. | aiohttp Response object | Tests that the function returns an aiohttp Response object containing the sensor information page when the parameters supplied to this function are valid. |
|  | SEN-001 | Tester supplies a valid request to the function as well as a dictionary containing the following values: {groupid: 0, sensorid: 0, alias: “abc”} | http://<base\_url>/sensors/info.html?groupid=0&sensorid=0&alias=abc  Where base\_url is one of localhost, 127.0.0.1, 192.168.xxx.xxx or domain. | Tests that the function returns a URL route for a sensor when all parameters are valid. |
|  | SEN-002 | Tester supplies a valid request to the function. | aiohttp Response object | Tests that the function returns an aiohttp Response object for the sensor listings page when the parameters supplied to this function are valid. |
|  | SOC-000 | Tester calls the function supplying a dictionary containing the tuple (0, 0), 0 for the groupid parameter, and 0 for the sensorid parameter. | True | Tests that the \_does\_room\_exist function returns True when all supplied parameters are valid, and the indicated room exists. |
|  | SOC-000 | Tester calls the function supplying an empty dictionary for the rooms parameter, 0 for the groupid parameter, and 0 for the sensorid parameter. | False | Tests that the \_does\_room\_exist function returns False when all supplied parameters are valid, and the indicated room does not exist. |
|  | SOC-000 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request requires 'groupid' field!”} | Tests that the \_does\_room\_exist function returns an error message and status code inside the body of an aiohttp Response when groupid is not included in the request. |
|  | SOC-000 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: -1, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter 'groupid' must be >= 0.”} | Tests that the \_does\_room\_exist function returns an error message and status code inside the body of an aiohttp Response when groupid is less than 0. |
|  | SOC-000 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: 1, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “"ERROR: No such group provisioned into the system!"”} | Tests that the \_does\_room\_exist function returns an error message and status code inside the body of an aiohttp Response when the group does not exist. |
|  | SOC-000 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request requires 'sensorid' field!”} | Tests that the \_does\_room\_exist function returns an error message and status code inside the body of an aiohttp Response when sensorid is not included in the request. |
|  | SOC-000 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: 0, “sensorid”: -1, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter 'sensorid' must be >= 0.”} | Tests that the \_does\_room\_exist function returns an error message and status code inside the body of an aiohttp Response when sensorid is less than 0. |
|  | SOC-000 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: 0, “sensorid”: 1, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “"ERROR: No such sensor provisioned into the system!"”} | Tests that the \_does\_room\_exist function returns an error message and status code inside the body of an aiohttp Response when the sensor does not exist. |
|  | SOC-001 | Tester calls the function passing in a dictionary for the rooms parameter, 0 for the groupid, 0 for the sensorid, and a WebSocket for ws. | True | Tests that the \_does\_ws\_exist function returns True when all parameters are valid, and the WebSocket exists. |
|  | SOC-001 | Tester calls the function passing in a dictionary for the rooms parameter, 0 for the groupid, 0 for the sensorid, and None for ws. | False | Tests that the \_does\_ws\_exist function returns False when all parameters are valid, and the WebSocket does not exist. |
|  | SOC-001 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_CLOSE”, “sensorid”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request requires 'groupid' field!”} | Tests that the \_does\_ws\_exist function returns an error message and status code inside the body of an aiohttp Response when groupid is not included in the request. |
|  | SOC-001 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_CLOSE”, “groupid”: -1, “sensorid”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter 'groupid' must be >= 0.”} | Tests that the \_does\_ws\_exist function returns an error message and status code inside the body of an aiohttp Response when groupid is less than 0. |
|  | SOC-001 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_CLOSE”, “groupid”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request requires 'sensorid' field!”} | Tests that the \_does\_ws\_exist function returns an error message and status code inside the body of an aiohttp Response when sensorid is not included in the request. |
|  | SOC-001 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_CLOSE”, “groupid”: 0, “sensorid”: -1} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter 'sensorid' must be >= 0.”} | Tests that the \_does\_ws\_exist function returns an error message and status code inside the body of an aiohttp Response when sensorid is less than 0. |
|  | SOC-001 |  | {“cmd”: “RESP\_ERROR”, “error”: “"ERROR: No such sensor provisioned into the system!"”} | Tests that the \_does\_ws\_exist function returns an error message and status code inside the body of an aiohttp Response when the sensor does not exist. |
|  | SOC-002 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_CLOSE”, “sensorid”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request requires 'groupid' field!”} | Tests that the \_leave function returns an error message and status code inside the body of an aiohttp Response when groupid is not included in the request. |
|  | SOC-002 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_CLOSE”, “groupid”: -1, “sensorid”: 0}. | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter 'groupid' must be >= 0.”} | Tests that the \_leave function returns an error message and status code inside the body of an aiohttp Response when groupid is less than 0. |
|  | SOC-002 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_CLOSE”, “groupid”: 1, “sensorid”: 0}. | {“cmd”: “RESP\_ERROR”, “error”: “"ERROR: No such group provisioned into the system!"”} | Tests that the \_leave function returns an error message and status code inside the body of an aiohttp Response when the group does not exist. |
|  | SOC-002 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_CLOSE”, “groupid”: 0}. | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request requires 'sensorid' field!”} | Tests that the \_leave function returns an error message and status code inside the body of an aiohttp Response when sensorid is not included in the request. |
|  | SOC-002 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_CLOSE”, “groupid”: 0, “sensorid”: -1}. | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter 'sensorid' must be >= 0.”} | Tests that the \_leave function returns an error message and status code inside the body of an aiohttp Response when sensorid is less than 0. |
|  | SOC-002 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_CLOSE”, “groupid”: 0, “sensorid”: 1}. | {“cmd”: “RESP\_ERROR”, “error”: “"ERROR: No such sensor provisioned into the system!"”} | Tests that the \_leave function returns an error message and status code inside the body of an aiohttp Response when the sensor does not exist. |
|  | SOC-003 | The tester uses the navigates through the home page to the sensor listing page and finally to the sensor information page using the Senslify web client. | The live chart is displayed and is animated. | Tests that the WebSocket is part of the indicated room after the \_join function executes. |
|  | SOC-003 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_JOIN”, “sensorid”: 0}. | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request requires 'groupid' field!”} | Tests that the \_join function returns False and error message in the body of an aiohttp Response object when the group identifier is not included in the request. |
|  | SOC-003 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_JOIN”, “groupid”: -1, “sensorid”: 0}. | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter 'groupid' must be >= 0.”} | Tests that the \_join function returns False and an error message in the body of an aiohttp Response object when the group identifier is less than 0. |
|  | SOC-003 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_JOIN”, “groupid”: 1, “sensorid”: 0}. | {“cmd”: “RESP\_ERROR”, “error”: “"ERROR: No such group provisioned into the system!"”} | Tests that the \_join function returns False and an error message in the body of an aiohttp Response object when the group does not exist. |
|  | SOC-003 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_JOIN”, “groupid”: 0}. | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request requires 'sensorid' field!”} | Tests that the \_join function returns False and an error message in the body of an aiohttp Response object when the sensor identifier is not provided. |
|  | SOC-003 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_JOIN”, “groupid”: 0, “sensorid”: -1}. | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter 'sensorid' must be >= 0.”} | Tests that the \_join function returns False and an error message in the body of an aiohttp Response object when the sensor identifier less than 0. |
|  | SOC-003 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_JOIN”, “groupid”: 0, “sensorid”: 1}. | {“cmd”: “RESP\_ERROR”, “error”: “"ERROR: No such sensor provisioned into the system!"”} | Tests that the \_join function returns False and an error message in the body of an aiohttp Response object when the sensor does not exist. |
|  | SOC-004 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_STREAM”, “rtypeid”: 1}. The tester must additionally already be on the sensor information page for sensorid 0, groupid 0 before conducting this test. | The sensor information page is now setup to receive sensor readings for rtypeid 1. | Tests that the WebSocket’s stream is changed after the function executes when all parameters are valid. |
|  | SOC-004 | The tester submits the following command over the client WebSocket from the JS console in their web browser: {“cmd”: “RQST\_STREAM”, “rtypeid”: -1}. The tester must additionally already be on the sensor information page for sensorid 0, groupid 0 before conducting this test. | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter ‘rtypeid’ must be >= 0!”} | Tests that the \_change\_stream function returns an error message and status code inside the body of an aiohttp Response when the reading type is invalid. |
|  | SOC-005 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: 0, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0. The tester must additionally already be on the sensor information page for sensorid 0, groupid 0 before conducting this test. | The tester sees the reading appear on the sensor information page for sensorid 0, groupid 0. | Tests that subscribed clients receive the broadcasted message when all parameters are valid. |
|  | SOC-005 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request requires 'groupid' field!”} | Tests that the message function returns an error message and status code inside the body of an aiohttp Response when the group identifier is not included in an uploaded message. |
|  | SOC-005 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: -1, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter 'groupid' must be >= 0.”} | Tests that the message function returns an error message and status code inside the body of an aiohttp Response when the group identifier is less than 0 in an uploaded message. |
|  | SOC-005 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: 1, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “"ERROR: No such group provisioned into the system!"”} | Tests that the message function returns an error message and status code inside the body of an aiohttp Response when the group specified in an uploaded message does not exist. |
|  | SOC-005 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request requires 'sensorid' field!”} | Tests that the message function returns an error message and status code inside the body of an aiohttp Response when the sensor identifier is not included in an uploaded message. |
|  | SOC-005 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: 0, “sensorid”: -1, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request parameter 'sensorid' must be >= 0.”} | Tests that the message function returns an error message and status code inside the body of an aiohttp Response when the sensor identifier is less than 0 in an uploaded message. |
|  | SOC-005 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: 0, “sensorid”: 1, “rtypeid”: 0, “val”: 50, “ts”: 0} | {“cmd”: “RESP\_ERROR”, “error”: “"ERROR: No such sensor provisioned into the system!"”} | Tests that the message function returns an error message and status code inside the body of an aiohttp Response when the sensor specified in an uploaded message does not exist. |
|  | SOC-005 | The tester submits the following command to the servers REST endpoint from the JS console in their web browser:  {“cmd”: “REST\_UPLOAD”, “groupid”: 0, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 0] | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: Request is not a properly formed JSON message!”} | Tests that the message function returns an error message and status code inside the body of an aiohttp Response when the message uploaded by a sensor is invalid. |
|  | SOC-006 | The tester submits the following command over the client WebSocket from the JS console in their web browser while on the sensor information page:  {“cmd”: “RQST\_STREAM”, “rtypeid”: 1} | {“cmd”: “RESP\_STREAM”, “readings”:[]} | Tests that the function properly routes a received WebSocket command when all parameters are valid. |
|  | SOC-006 | The tester submits the following command over the client WebSocket from the JS console in their web browser while on the sensor information page:  {“cmd”: “RQST\_PAUSE”} | {“cmd”: “RESP\_ERROR”, “error”: “ERROR: ‘cmd’ must be one of {‘RQST\_JOIN’, ‘RQST\_CLOSE’, ‘RQST\_STREAM’, ‘RQST\_SENSOR\_STATS’, ‘RQST\_DOWNLOAD’}!”} | Tests that an error is returned to the client WebSocket when a command parameter is invalid. |
|  | SOC-007 | The tester shuts down the web server while at least one client is connected to the web server via a WebSocket. | N/A | Tests that all WebSockets are gracefully shutdown server-side when the web server shuts down. |
|  | VER-000 | Tester supplies a valid request and a Python dictionary containing the following mappings to this function: {“target”: ”groups”}. | True, None | Tests that the \_verify\_find\_request function returns True and None when all parameters are valid. |
|  | VER-000 | Tester supplies a valid request as well as a None type value as the params parameter to this function. | False, “ERROR: Request parameters must not be null!” | Tests that the \_verify\_find\_request function returns False and an error message when the params dictionary is invalid. |
|  | VER-000 | Tester supplies a valid request as well as an empty Python list to this function. | False, “ERROR: Request parameters must be a JSON object!” | Tests that the \_verify\_find\_request function returns False and an error message when the params parameter is not a dictionary. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {}. | False, “ERROR: Request params requires ‘target’ field!” | Tests that the \_verify\_find\_request function returns False and an error message when the target parameter is not specified by the params dictionary. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: ”fish”}. | False, “ERROR: Request parameter ‘target’ must be one of {‘groups’, ‘rtypes’, ‘sensors’, ‘readings’}!” | Tests that the \_verify\_find\_request function returns False and an error message when an invalid target is specified by the params dictionary. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensors”}. | False, “ERROR: Request params requires ‘groupid’ field!” | Tests that the \_verify\_find\_request function returns False and an error message when the target is sensors and the groupid parameter is not specified by the params dictionary. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensors”, “groupid”: “abc”}. | False, “ERROR: A parameter is of incorrect type!” | Tests that the \_verify\_find\_request function returns False and an error message when the target is sensors and a parameter specified by the params dictionary is not of the correct type. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensors”, “groupid”: -1}. | False, “ERROR: Request parameter ‘groupid’ must be >= 0” | Tests that the \_verify\_find\_request function returns False and an error message when the target is sensors and the groupid parameter is less than 0. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensors”, “groupid”: 1}. | False, “ERROR: No such group provisioned into the system!” | Tests that the \_verify\_find\_request function returns False and an error message when the target is sensors and the group specified by the params dictionary is not provisioned into the systems database. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “readings”, “sensorid”: 0}. | False, “ERROR: Request params requires ‘groupid’ field!” | Tests that the \_verify\_find\_request function returns False and an error message when the target is readings and the groupid parameter is not specified by the params dictionary. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “readings”, “grouopid”: 0}. | False, “ERROR: Request params requires ‘sensorid’ field!” | Tests that the \_verify\_find\_request function returns False and an error message when the target is readings and the sensorid parameter is not specified by the params dictionary. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “readings”, “groupid”: “abc”, “sensorid”: 0}. | False, “ERROR: A parameter is on incorrect type!” | Tests that the \_verify\_find\_request function returns False and an error message when the target is readings and a parameter is not of the correct type. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “readings”, “groupid”: -1, “sensorid”: 0}. | False, “ERROR: Request parameter ‘groupid’ must be >= 0!” | Tests that the \_verify\_find\_request function returns False and an error message when the target is readings and the groupid parameter is less than 0. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “readings”, “groupid”: 0, “sensorid”: -1}. | False, “ERROR: Request parameter ‘sensorid’ must be >= 0!” | Tests that the \_verify\_find\_request function returns False and an error message when the target is readings and the sensorid parameter is less than 0. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “readings”, “groupid”: 1, “sensorid”: 0}. | False, “ERROR: No such group provisioned into the system!” | Tests that the \_verify\_find\_request function returns False and an error message when a the target is readings, a group is specified by the params dictionary, and the group is not provisioned in the system database. |
|  | VER-000 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “readings”, “groupid”: 1, “sensorid”: 0}. | False, “ERROR: No such sensor provisioned into the system!” | Tests that the \_verify\_find\_request function returns False and an error message when the target is readings, a sensor is specified by the params dictionary, and the sensor is not provisioned into the system database. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “group”, “groupid”: 0, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | True, None | Tests that the \_verify\_stats\_request function return True and None when all supplied parameters are valid. |
|  | VER-001 | Tester supplies a valid request as well as a None type value as the params parameter to this function. | False, “ERROR: Request parameters must not be null!” | Tests that the \_verify\_stats\_request function returns False and an error message when the params dictionary is invalid. |
|  | VER-001 | Tester supplies a valid request as well as an empty Python list to this function. | False, “ERROR: Request parameters must be a JSON object!” | Tests that the \_verify\_stats\_request function returns False and an error message when the params parameter is not a dictionary. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: 0, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request params requires ‘target’ field!” | Tests that the \_verify\_stats\_request function returns False and an error message when the target parameter is not specified by the params dictionary. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “group”, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request params requires ‘groupid’ field!” | Tests that the \_verify\_stats\_request function returns False and an error message when the groupid parameter is not specified by the params dictionary. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “group”, “groupid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request params requires ‘rtypeid’ field!” | Tests that the \_verify\_stats\_request function returns False and an error message when the rtypeid parameter is not specified by the params dictionary. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “group”, “groupid”: 0, “rtypeid”: 0, “end\_ts”: 1625859638}. | False, “ERROR: Request params requires ‘start\_ts’ field!” | Tests that the \_verify\_stats\_request function returns False and an error message when the start\_ts parameter is not specified by the params dictionary. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “group”, “groupid”: 0, “rtypeid”: 0, “start\_ts”: 1625857799}. | False, “ERROR: Request params requires ‘end\_ts’ field!” | Tests that the \_verify\_stats\_request function returns False and an error message when the end\_ts field is not specified by the params dictionary. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “groups”, “groupid”: 0, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request parameter ‘target’ must be one of {‘group’, ‘sensor’}!” | Tests that the \_verify\_stats\_request function returns False and an error message when the target parameter is not a valid value. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “groupid”: 0, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request params requires ‘sensorid’ field!” | Tests that the \_verify\_stats\_request function returns False and an error message when the target is sensors and the sensorid parameter is not included in the params dictionary. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “sensorid”: -1, “groupid”: 0, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request parameter ‘sensorid’ must be >= 0!” | Tests that the \_verify\_stats\_request function returns False and an error message when the sensorid parameter is less than 0. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “sensorid”: ”abc”, “groupid”: 0, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: A parameter is of incorrect type!” | Tests that the \_verify\_stats\_request function returns False and an error message when a parameter specified by the params dictionary is not of the correct type. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “sensorid”: 0, “groupid”: -1, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request parameter ‘groupid’ must be >= 0!” | Tests that the \_verify\_stats\_request function returns False and an error message when the groupid parameter is less than 0. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “sensorid”:0, “groupid”: 0, “rtypeid”: -1, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request parameter ‘rtypeid’ must be >= 0!” | Tests that the \_verify\_stats\_request function returns False and an error message when the rtypeid parameter is less than 0. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “sensorid”:0, “groupid”: 0, “rtypeid”: 0, “start\_ts”: -1, “end\_ts”: 1625859638}. | False, “ERROR: Request parameter ‘start\_ts’ must be >= 0!” | Tests that the \_verify\_stats\_request function returns False and an error message when the start\_ts parameter is less than 0. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “sensorid”:0, “groupid”: 0, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: -1}. | False, “ERROR: Request parameter ‘end\_ts’ must be >= 0!” | Tests that the \_verify\_stats\_request function returns False and an error message when the end\_ts parameter is less than 0. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625857799}. | False, “ERROR: Request parmeter 'start\_ts must be < ''end\_ts!” | Tests that the \_verify\_stats\_request function returns False and an error message when the start\_ts parameter is equal to the end\_ts parameter. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 0, “start\_ts”: 1625900000, “end\_ts”: 1625857799}. | False, “ERROR: Request parmeter 'start\_ts must be < ''end\_ts!” | Tests that the \_verify\_stats\_request function returns False and an error message when the start\_ts parameter is greater than the end\_ts parameter. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “sensorid”:0, “groupid”: 1, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: No such group provisioned into the system!” | Tests that the \_verify\_stats\_request function returns False and an error message when the group specified by the parameters in the params dictionary is not provisioned into the systems database. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “sensorid”:0, “groupid”: 1, “rtypeid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: No such sensor provisioned into the system!” | Tests that the \_verify\_stats\_request function returns False and an error message when the target is sensors and the sensor specified by the parameters in the params dictionary is not provisioned into the systems database. |
|  | VER-001 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “sensorid”:0, “groupid”: 0, “rtypeid”: 6, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: No such reading type provisioned into the system!” | Tests that the \_verify\_stats\_request function returns False and an error message when the reading type specified by the parameters in the params dictionary is not provisioned into the systems database. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | True, None | Tests that the \_verify\_download\_request function returns True and None when all supplied parameters are valid. |
|  | VER-002 | Tester supplies a valid request as well as a None type value as the params parameter to this function. | False, “ERROR: Request parameters must not be null!” | Tests that the \_verify\_download\_request function returns False and an error message when the params dictionary is invalid. |
|  | VER-002 | Tester supplies a valid request as well as an empty Python list to this function. | False, “ERROR: Request parameters must be a JSON object!” | Tests that the \_verify\_download\_request function returns False and an error message when the params parameter is not a dictionary. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: { “groupid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request params requires ‘sensorid’ field!” | Tests that the \_verify\_download\_request function returns False and an error message when the sensorid parameter is not included in the params dictionary. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request params requires ‘groupid’ field!” | Tests that the \_verify\_download\_request function returns False and an error message when the groupid parameter is not included in the params dictionary. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 0, “end\_ts”: 1625859638}. | False, “ERROR: Request params requires ‘start\_ts’ field!” | Tests that the \_verify\_download\_request function returns False and an error message when the start\_ts parameter is not included in the params dictionary. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 0, “start\_ts”: 1625857799}. | False, “ERROR: Request params requires ‘end\_ts’ field!” | Tests that the \_verify\_download\_request function returns False and an error message when the end\_ts parameter is not included in the params dictionary. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: “abc”, “groupid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: A parameter is of incorrect type!” | Tests that the \_verify\_download\_request function returns False and an error message when a required parameter from the params dictionary is not of the correct type. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: -1, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request parameter ‘groupid’ must be >= 0!” | Tests that the \_verify\_download\_request function returns False and an error message when the groupid parameter is less than 0. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: -1, “groupid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: Request parameter ‘sensorid’ must be >= 0!” | Tests that the \_verify\_download\_request function returns False and an error message when the sensorid parameter is less than 0. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 0, “start\_ts”: -1, “end\_ts”: 1625859638}. | False, “ERROR: Request parameter ‘start\_ts’ must be >= 0!” | Tests that the \_verify\_download\_request function returns False and an error message when the start\_ts parameter is less than 0. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 0, “start\_ts”: 1625857799, “end\_ts”: -1}. | False, “ERROR: Request parameter ‘end\_ts’ must be >= 0!” | Tests that the \_verify\_download\_request function returns False and an error message when the end\_ts parameter is less than 0. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 0, “start\_ts”: 1625857799, “end\_ts”: 1625857799}. | False, “ERROR: Request parmeter 'start\_ts must be < ''end\_ts!” | Tests that the \_verify\_download\_request function returns False and an error message when the start\_ts parameter is equal to the end\_ts parameter. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 0, “start\_ts”: 1625900000, “end\_ts”: 1625857799}. | False, “ERROR: Request parmeter 'start\_ts must be < ''end\_ts!” | Tests that the \_verify\_download\_request function returns False and an error message when the start\_ts parameter is greater than the end\_ts parameter. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 1, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: No such group provisioned into the system!” | Tests that the \_verify\_download\_request function returns False and an error message when the group specified by the parameters in the params dictionary is not provisioned into the systems database. |
|  | VER-002 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0, “groupid”: 1, “start\_ts”: 1625857799, “end\_ts”: 1625859638}. | False, “ERROR: No such sensor provisioned into the system!” | Tests that the \_verify\_download\_request function returns False and an error message when the sensor specified by the parameters in the params dictionary is not provisioned into the systems database. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 0, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 1625857799}]}. | True, None | Tests that the \_verify\_upload\_request function returns True and None when all supplied parameters are valid. |
|  | VER-003 | Tester supplies a valid request as well as a None type value as the params parameter to this function. | False, “ERROR: Request parameters must not be null!” | Tests that the \_verify\_upload\_request function returns False and an error message when the params dictionary is invalid. |
|  | VER-003 | Tester supplies a valid request as well as an empty Python list to this function. | False, “ERROR: Request parameters must be a JSON object!” | Tests that the \_verify\_upload\_request function returns False and an error message when the params parameter is not a dictionary. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {}. | False, “ERROR: Request params requires ‘readings’ field!” | Tests that the \_verify\_upload\_request function returns False and an error message when the readings parameter is not present in the params dictionary. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: {}}. | False, “ERROR: Request parameter ‘readings’ must be a JSON array!” | Tests that the \_verify\_upload\_request function returns False and an error message when the readings parameter is not a list. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 1625857799}]}. | False, “ERROR: Request params requires ‘groupid’ field!” | Tests that the \_verify\_upload\_request function returns False and an error message when the groupid parameter is not included in the params dictionary. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 1625857799}]}. | False, “ERROR: Request params requires ‘sensorid’ field!” | Tests that the \_verify\_upload\_request function returns False and an error message when the sensorid parameter is not included in the params dictionary. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 0, “sensorid”: 0, “val”: 50, “ts”: 1625857799}]}. | False, “ERROR: Request params requires ‘rtypeid’ field!” | Tests that the \_verify\_upload\_request function returns False and an error message when the rtypeid parameter is not included in the params dictionary. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 0, “sensorid”: 0, “rtypeid”: 0, “ts”: 1625857799}]}. | False, “ERROR: Request params requires ‘val’ field!” | Tests that the \_verify\_upload\_request function returns False and an error message when the val parameter is not included in the params dictionary. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 0, “sensorid”: 0, “rtypeid”: 0, “val”: 50}]}. | False, “ERROR: Request params requires ‘ts’ field!” | Tests that the \_verify\_upload\_request function returns False and an error message when the ts parameter is not included in the params dictionary. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: “abc”, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 1625857799}]}. | False, “ERROR: A parameter is of incorrect type!” | Tests that the \_verify\_upload\_request function returns False and an error message when a required parameter from the params dictionary is not of the correct type. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: -1, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 1625857799}]}. | False, “ERROR: Request parameter ‘groupid’ must be >= 0!” | Tests that the \_verify\_upload\_request function returns False and an error message when the groupid parameter is less than 0. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 0, “sensorid”: -1, “rtypeid”: 0, “val”: 50, “ts”: 1625857799}]}. | False, “ERROR: Request parameter ‘sensorid’ must be >= 0!” | Tests that the \_verify\_upload\_request function returns False and an error message when the sensorid parameter is less than 0. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 0, “sensorid”: 0, “rtypeid”: -1, “val”: 50, “ts”: 1625857799}]}. | False, “ERROR: Request parameter ‘rtypeid’ must be >= 0!” | Tests that the \_verify\_upload\_request function returns False and an error message when the rtypeid parameter is less than 0. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 0, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: -1}]}. | False, “ERROR: Request parameter ‘ts’ must be >= 0!” | Tests that the \_verify\_upload\_request function returns False and an error message when the ts parameter is less than 0. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 1, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 1625857799}]}. | False, “ERROR: No such group provisioned into the system!” | Tests that the \_verify\_upload\_request function returns False and an error message when the group specified by the parameters in the params dictionary is not already provisioned into the systems database. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 1, “sensorid”: 0, “rtypeid”: 0, “val”: 50, “ts”: 1625857799}]}. | False, “ERROR: No such sensor provisioned into the system!” | Tests that the \_verify\_upload\_request function returns False and an error message when the sensor specified by the parameters in the params dictionary is not already provisioned into the systems database. |
|  | VER-003 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“readings”: [{“groupid”: 0, “sensorid”: 0, “rtypeid”: 6, “val”: 50, “ts”: 1625857799}]}. | False, “ERROR: No such reading type provisioned into the system!” | Tests that the \_verify\_upload\_request function returns False and an error message when the reading type specified by the parameters in the params dictionary is not already provisioned into the systems database. |
|  | VER-004 | Tester supplies a valid request as well as the following Python dictionary to this function: {“target”: “group”, “alias”: “TEST-0261”} | True, None | Tests that the \_verify\_provision\_command function return True when all supplied parameters are valid. |
|  | VER-004 | Tester supplies a valid request as well as a None type value as the params parameter to this function. | False, “ERROR: Request parameters must not be null!” | Tests that the \_verify\_provision\_command function returns False and an error message when the params dictionary is invalid. |
|  | VER-004 | Tester supplies a valid request as well as an empty Python list to this function. | False, “ERROR: Request parameters must be a JSON object!” | Tests that the \_verify\_provision\_command function returns False and an error message when the params parameter is not a dictionary. |
|  | VER-004 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“alias”: “TEST-0265”}. | False, “ERROR: Request params requires ‘target’ field!” | Tests that the \_verify\_provision\_command function returns False and an error message when the target field specified by the params dictionary is not included in the params dictionary. |
|  | VER-004 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “rtypeid”, “alias”: “TEST-0266”}. | False, “ERROR: Invalid ‘target’ specified! Must be one of {‘sensor’, ‘group’}.” | Tests that the \_verify\_provision\_command function returns False and an error message when an invalid provisioning target is specified by the params dictionary. |
|  | VER-004 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “alias”: “TEST-0267”}. | False, “ERROR: Request params requires ‘groupid’ field!” | Tests that the \_verify\_provision\_command function returns False and an error message when the groupid field specified by the params dictionary is not included in the params dictionary. |
|  | VER-004 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “groupid”: “abc”, “alias”: “TEST-0268”}. | False, “ERROR: Request parameter ‘groupid’ must be an integer!” | Tests that the \_verify\_provision\_command function returns False and an error message when a sensor is provisioned and the groupid field specified by the params dictionary is not an integer. |
|  | VER-004 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “groupid”: -1, “alias”: “TEST-0269”}. | False, “ERROR: Request parameter ‘groupid’ must be >= 0!” | Tests that the \_verify\_provision\_command function returns False and an error message when a sensor is provisioned and the groupid field specified by the params dictionary is less than 0. |
|  | VER-004 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“target”: “sensor”, “groupid”: 0, “alias”: “”}. | False, “ERROR: Request parameter ‘alias’ must contain at least one (1) character!” | Tests that the \_verify\_provision\_command function returns False and an error message when the optional alias parameter specified by the params dictionary is less than one character in length. |
|  | VER-005 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function:  {“groupid”: 0, “sensorid”: 0} | True, None | Tests that the \_verify\_join\_command function return True when all supplied parameters are valid. |
|  | VER-005 | Tester supplies a valid request as well as a None type value as the params parameter to this function. | False, “ERROR: Request parameters must not be null!” | Tests that the \_verify\_join\_command function returns False and an error message when the params dictionary is invalid. |
|  | VER-005 | Tester supplies a valid request as well as an empty Python list to this function. | False, “ERROR: Request parameters must be a JSON object!” | Tests that the \_verify\_join\_command function returns False and an error message when the params dictionary is not a dictionary. |
|  | VER-005 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0}. | False, “ERROR: Request requires ‘groupid’ field!” | Tests that the \_verify\_join\_command function returns False and an error message when the groupid field is not included in the params dictionary. |
|  | VER-005 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: 0}. | False, “ERROR: Request requires ‘sensorid’ field!” | Tests that the \_verify\_join\_command function returns False and an error message when the sensorid field is not included in the params dictionary. |
|  | VER-005 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: ”abc”, “sensorid”: 0}. | False, “ERROR: A parameter is of incorrect type!” | Tests that the \_verify\_join\_command function returns False and an error message when a close command parameter is not the correct type. |
|  | VER-005 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: -1, “sensorid”: 0}. | False, “ERROR: Request parameter ‘groupid’ must be >= 0.” | Tests that the \_verify\_join\_command function returns False and an error message when the groupid field specified in the params dictionary is less than 0. |
|  | VER-005 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: 0, “sensorid”: -1}. | False, “ERROR: Request parameter ‘sensorid’ must be >= 0.” | Tests that the \_verify\_join\_command function returns False and an error message when the sensorid field specified in the params dictionary is less than 0. |
|  | VER-005 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: 1, “sensorid”: 0}. | False, “ERROR: No such group provisioned into the system!” | Tests that the \_verify\_join\_command function returns False and an error message when the group specified by the request parameters does not exist. |
|  | VER-005 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: 0, “sensorid”: 1}. | False, “ERROR: No such sensor provisioned into the system!” | Tests that the \_verify\_join\_command function returns False and an error message when the sensor specified by the request parameters does not exist. |
|  | VER-006 | Tester supplies a valid request as well as the following Python dictionary to this function: {“groupid”: 0, “sensorid”: 0}. | True, None | Tests that the \_verify\_close\_command function return True when all supplied parameters are valid. |
|  | VER-006 | Tester supplies a valid request as well as a None type value as the params parameter to this function. | False, “ERROR: Request parameters must not be null!” | Tests that the \_verify\_close\_command function returns False and an error message when the params dictionary is invalid. |
|  | VER-006 | Tester supplies a valid request as well as an empty Python list to this function. | False, “ERROR: Request parameters must be a JSON object!” | Tests that the \_verify\_close\_command function returns False and an error message when the params parameter is not a dictionary. |
|  | VER-006 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“sensorid”: 0}. | False, “ERROR: Request requires ‘groupid’ field!” | Tests that the \_verify\_close\_command function returns False and an error message when the groupid field is not included in the params dictionary. |
|  | VER-006 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: 0}. | False, “ERROR: Request requires ‘sensorid’ field!” | Tests that the \_verify\_close\_command function returns False and an error message when the sensorid field is not included in the params dictionary. |
|  | VER-006 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: “abc”, “sensorid”: 0}. | False, “ERROR: A parameter is of incorrect type!” | Tests that the \_verify\_close\_command function returns False and an error message when a close command parameter is not the correct type. |
|  | VER-006 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: -1, “sensorid”: 0}. | False, “ERROR: Request parameter ‘groupid’ must be >= 0.” | Tests that the \_verify\_close\_command function returns False and an error message when the groupid field specified in the params dictionary is less than 0. |
|  | VER-006 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: 0, “sensorid”: -1}. | False, “ERROR: Request parameter ‘sensorid’ must be >= 0.” | Tests that the \_verify\_close\_command function returns False and an error message when the sensorid field specified in the params dictionary is less than 0. |
|  | VER-006 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: 1, “sensorid”: 0}. | False, “ERROR: No such group provisioned into the system!” | Tests that the \_verify\_close\_command function returns False and an error message when the group specified by the request parameters does not exist. |
|  | VER-006 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“groupid”: 1, “sensorid”: 1}. | False, “ERROR: No such sensor provisioned into the system!” | Tests that the \_verify\_close\_command function returns False and an error message when the sensor specified by the request parameters does not exist. |
|  | VER-007 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“rtypeid”: 0}. | True, None | Tests that the \_verify\_stream\_command function returns True when all parameters are valid. |
|  | VER-007 | Tester supplies a valid request as well as a None type value as the params parameter to this function. | False, “ERROR: Request parameters must not be null!” | Tests that the \_verify\_stream\_command function returns False and an error message when the params dictionary is invalid. |
|  | VER-007 | Tester supplies a valid request as well as an empty Python list to this function. | False, “ERROR: Request parameters must be a JSON object!” | Tests that the \_verify\_stream\_command function returns False and an error message when the params parameter is not a dictionary. |
|  | VER-007 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {}. | False, “ERROR: Request requires ‘rtypeid’ field!” | Tests that the \_verify\_stream\_command function returns False and an error message when a the rtypeid parameter is not included in the params dictionary. |
|  | VER-007 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“rtypeid”: “abc”}. | False, “ERROR: A parameter is of incorrect type!” | Tests that the \_verify\_stream\_command function returns False and an error message when a parameter included in the params dictionary is not the correct type. |
|  | VER-007 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“rtypeid”: -1}. | False, “ERROR: Request parameter ‘rtypeid’ must be >= 0!” | Tests that the \_verify\_stream\_command function returns False and an error message when the rtypeid parameter is less than 0. |
|  | VER-007 | Tester supplies a valid request as well as a Python dictionary containing the following mappings to this function: {“rtypeid”: 6}. | False, “ERROR: No such reading type provisioned into the system!” | Tests that the \_verify\_stream\_command function returns False and an error message when the rtypeid parameter does not exist in the system database. |
|  | VER-008 | Tester supplies the following JSON object as a serialized string inside the request body passed to this function: {“cmd”: “provision”, “params”: {“target”: “group”, “alias”: “CSU-WH”}}. | True, None | Tests that the verify\_rest\_request function returns True when all parameters are valid. |
|  | VER-008 | Tester supplies the following JSON object as a serialized string inside the request body passed to this function: {“params”: {“target”: “group”}}. | False, “ERROR: Request requires ‘cmd’ field!” | Tests that the verify\_rest\_request function returns False and an error message when a rest command is not specified in the request. |
|  | VER-008 | Tester supplies the following JSON object as a serialized string inside the request body passed to this function: {“cmd”: “search”, “params”: {“target”: “group”}}. | False, “ERROR: ‘cmd’ must be one of {‘find’, ‘stats’, ‘download’, ‘upload’, ‘provision’}!” | Tests that the verify\_rest\_request function returns False and an error message when an invalid rest command is specified in the request. |
|  | VER-008 | Tester supplies the following JSON object as a serialized string inside the request body passed to this function: {“cmd”: “provision”}. | False, “ERROR: Request requires ‘params’ field!” | Tests that the verify\_rest\_request function returns False and an error message when the params dictionary is not specified in the request. |
|  | VER-009 | Tester supplies a valid request as well as the following JSON object as a serialized string inside the request body passed to this function: {“cmd”: “RQST\_JOIN”, “groupid”: 0,” sensorid”: 0}. | True, None | Tests that the verify\_ws\_request function returns True when all parameters are valid. |
|  | VER-009 | Tester supplies the following JSON object as a serialized string inside the request body passed to this function: {“groupid”: 0, “sensorid”: 0}. | False, “ERROR: Request requires ‘cmd’ field!” | Tests that the verify\_ws\_request function returns False and an error message when the request does not include a cmd parameter. |
|  | VER-009 | Tester supplies the following JSON object as a serialized string inside the request body passed to this function: {“cmd”: “RQST\_PAUSE”, “groupid”: 0, “sensorid”: 0}. | False, “ERROR: ‘cmd’ must be one of {‘RQST\_JOIN’, ‘RQST\_CLOSE’, ‘RQST\_STREAM’, ‘RQST\_SENSOR\_STATS’, ‘RQST\_DOWNLOAD’}!” | Tests that the verify\_ws\_request function returns False and an error message when the cmd is invalid. |