How to use the MK-AT API

Minimum Software Version 10.8.4

Features added from specific versions are indicated next to the method call definitions

Version 1.500

Page Intentionally Left Blank

MODIFICATIONS

Issue	Date	Modified by	Modified Pages	Observations
1.000	15/Dec/16	Andy Keys	All	First Release
1.1	16/Jan/17	Jason Pyke	All	Re-written to show actual API implementation
1.2	20/06/17	Jason Pyke	7,8	Added support for reading from external devices
1.3	25/05/18	Jason Pyke	9,10,11,14,15	Added support for running a test program via the API
1.4	26/07/18	David Watson	7,8	Added Load test program and delete test program
1.5	14/08/18	David Watson	11	Added validation error for user inputs

Contents

1 Purpose **1**

2 Configuring API Mode **2**

3 Testing The API **3**

4 API Methods **4**

4.1 StartContinuityTest – POST Method 4

4.1.1 Request Object/Parameters 4
4.1.2 Returns 4

4.2 StartShortCircuitTest – POST Method 4

4.2.1 Request Object/Parameters 4
4.2.2 Returns 5

4.3 StartHVInsulationTest – POST Method 5

4.3.1 Request Object/Parameters 5
4.3.2 Returns 6

4.4 StartHVHiPotTest – POST Method 6

4.4.1 Request Object/Parameters 6
4.4.2 Returns 6

4.5 RunExternalDeviceCommand – POST Method 6

4.5.1 Request Object/Parameters 7
4.5.2 Returns 7

4.6 LoadtestProgram – POST Method 7

4.6.1 Request Object/Parameters 7
4.6.2 Returns 8

4.7 DeleteAllTestPrograms – GET Method 8

4.7.1 Returns 8

4.8 GetSupportedDevices – GET Method 8

4.8.1 Returns 8

4.9 SwitchOffAllDevicesEMs – GET Method 9

4.9.1 Returns 9

4.10 GetSystemInformation – GET Method 9

4.10.1 Returns 9

4.11 GetLatestResult – GET Method 9

4.11.1 Parameters 10
4.11.2 Returns 10

4.12 GetSubTestResult – GET Method 10

4.12.1 Parameters 10
4.12.2 Returns 10
4.13 StopRunningTest – GET Method 10
4.13.1 Returns 10
4.14 GetTestStatus – GET Method 11
4.14.1 Returns 11
4.15 GetAllTestPrograms – GET Method 11
4.15.1 Returns 11
4.16 StartTestProgram – POST Method 11
4.16.1 Request Object/Parameters 12
4.16.2 Returns 12
5 Result Objects – Additional Information 13
5.1 StartTestProgramResult 13
5.2 StartTestResult 13
5.3 ContinuityTestResult 13
5.3.1 ConnectionResult 14
5.4 ShortCircuitTestResult 14
5.4.1 ShortCircuitNetListResult 14
5.5 HVHiPotResult 14
5.5.1 HVHiPotNetListResult 14
5.6 HVInsulationResult 15
5.6.1 HVInsulationNetListResult 15
5.7 StartTestProgrameResult 15
5.7.1 SubTestResult 15
5.7.2 AdvancedResult 16
6 Real-Time Updates 17
6.1 StatusChanged 17
6.2 TestCompleted 17
7 Expected Values 18
8 Examples 23
8.1 Via Javascript/SignalR 23
8.1.1 Important Notes 23
8.1.2 Example 1: To start a continuity test returning JSON: 23
8.1.3 Example 2: To start a continuity test returning XML: 23
8.1.4 Example 3: To listen for status updates: 24
8.1.5 Example 3: To retrieve the result ID 1001: 25

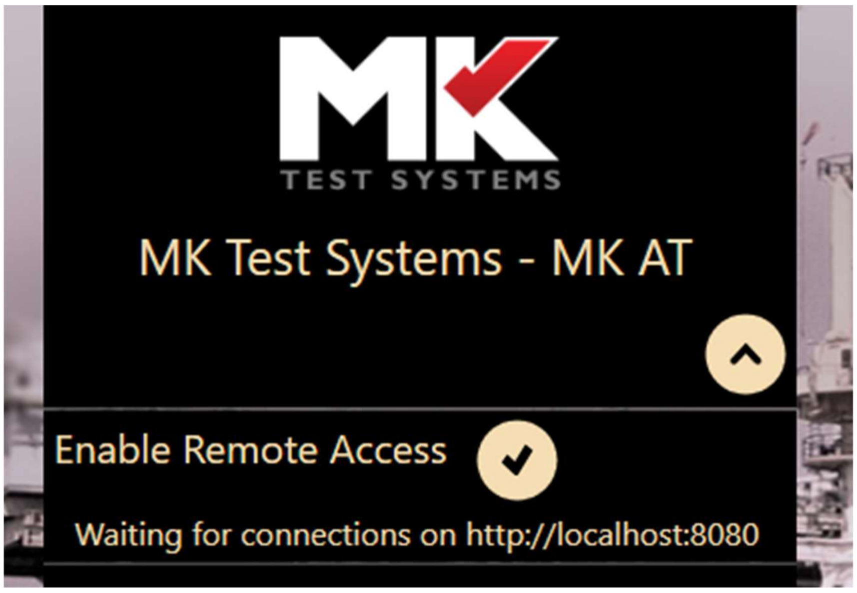
8.2 Via C# 26
8.2.1 Example 1: To start a continuity test: 26
8.2.2 Example 2: To listen for status updates: 26

1 Purpose

This document outlines, in technical detail, how to integrate with the MK AT API. The API will allow external applications to control the MK AT AutoMeg tester to take measurements without having to use the built-in software.

2 Configuring API Mode

To enable API mode you must be logged out of the software (on the log in screen) and click the down arrow near the top of the screen. This will give you a toggle “Enable Remote Access” to enable/disable API mode.



Waiting for connections on http://localhost:8080

You cannot use the API mode at the same time as being logged in to the MK AT AutoMeg.

There are additional options for configuring API mode within the "Administration" tab of the Configuration options when logged in to the MK AT AutoMeg.

Within this screen you can configure the startup options for the API as well as change the address or port which the API starts on.

Please note that if you want to use the MK AT AutoMeg across the network you can change the Remote Access Root URL to use a wildcard e.g. "http://*:8080", but for this to work you will need to exit the MK AT AutoMeg software and run it with administrator privileges. Alternatively, you can run MK AT in non-administrator mode, but you will need to register the port with Windows. To do this you will need to run a command windows as an administrator and run:

```
netsh http add urlacl url=http://*:8080/ user=Everyone
```

3 Testing The API

MK AT API mode comes with a built in test web page, to access it you can open the http://localhost:8080/MKATApiControlPanel path in your browser. Please note this web page only support IE 9 and above.

4 API Methods

The API is built around web standards using GET/POST requests as appropriate. Where a POST is used, the method will expect a populated object within the post body. This object must be in the format shown either as JSON or XML – you can set which by providing a content-type header.

4.1 StartContinuityTest – POST Method

Will start a continuity test and will return a status indicating whether the starting of the test was successful (it won't wait for the measurements to be taken before returning)

4.1.1 Request Object/Parameters

StartContinuityTestRequest

- Connections – {Collection<Connection>} – This defines the connections you wish to test. Each connection is made up of 2 parameters and each should be an integer value representing the test point number you wish to connect:
    - From – The test point to connect from
    - To – The test point to connect to
- VoltageLimitV - {double} - The maximum voltage in Volts that you wish to use for this test.
- CurrentmA - {double} - The current used to run the continuity test.
- DwellmS – {int} - The number of milliSeconds of Dwell you wish to apply after applying the current before the measurement is taken
- MaxMeasuredResistanceOhms - {int} - The maximum resistance you are interested in being reported by this test in Ohms (used to determine the minimum current for this test). The higher the value the longer a test may take to complete.
- SwitchOffDevicesEMsBeforeandAfter – {bool} – Indicates whether the EMS and external devices will be switched off before and after the test. Defaults to true.
- Mode – {TestPointMode} – The mode which the test should be run in. See the TestPointMode section for possible values

4.1.2 Returns

A StartTestResult object.

4.2 StartShortCircuitTest – POST Method

Will start a low voltage shorts test and will return a status indicating whether the starting of the test was successful (it won't wait for the measurements to be taken before returning)

4.2.1 Request Object/Parameters

StartShortCircuitTestRequest

- NetLists – {Collection<Connection>} – This defines the net lists you wish to test. Each net list is made up of a single “From” parameter, which should be a string value representing the test point range you wish to use as the “From Net” you wish to connect. The net list can use comma to delimit the test points, or hyphen to denote a range e.g. “1,2,3,5-9,12”. You must specify multiple of these to make a complete short circuit test, the API will test shorts across each of these net lists to the other net lists defined

- VoltageLimitV - {double} - The maximum voltage in Volts that you wish to use for this test.
- CurrentLimitmA - {double} - The maximum current which will be used to find the correct resistance when a short is found.
- MaxMeasuredResistanceOhms - {int} - The maximum resistance you are interested in being reported by this test in Ohms (used to determine the minimum current for this test). The higher the value the longer a test may take to complete.
- SwitchOffDevicesEMsBeforeandAfter – {bool} – Indicates whether the EMS and external devices will be switched off before and after the test. Defaults to true.
- Mode – {TestPointMode} – The mode which the test should be run in. See the TestPointMode section for possible values

4.2.2 Returns

A StartTestResult object.

4.3 StartHVInsulationTest – POST Method

Will start a High voltage DC insulation test and will return a status indicating whether the starting of the test was successful (it won't wait for the measurements to be taken before returning)

4.3.1 Request Object/Parameters

StartHVInsulationTestRequest

- NetLists – {Collection<Connection>} – This defines the net lists you wish to test. Each net list is made up of a single “From” parameter, which should be a string value representing the test point range you wish to use as the “From Net” you wish to connect. The net list can use comma to delimit the test points, or hyphen to denote a range e.g. “1,2,3,5-9,12”. You must specify multiple of these to make a complete short circuit test, the API will test shorts across each of these net lists to the other net lists defined
- VoltageV - {int} – The voltage to carry out the HV test at
- RampUpTimemS – {int} – The ramp up Time in milliseconds
- PreDwellTimemS – {int} – The time after ramp complete before measurement starts in milliseconds
- MeasureTimemS – {int} – The time in milliseconds to do the measurement for.
- RampDownTimemS – {int} – the time in seconds for the ramp down.
- SwitchOffDevicesEMsBeforeandAfter – {bool} – Indicates whether the EMS and external devices will be switched off before and after the test. Defaults to true.

- Mode – {TestPointMode} – The mode which the test should be run in. See the TestPointMode section for possible values

4.3.2 Returns

A StartTestResult object.

4.4 StartHVHiPotTest – POST Method

Will start a High voltage AC or DC Hi Pot test and will return a status indicating whether the starting of the test was successful (it won't wait for the measurements to be taken before returning)Will start a High

4.4.1 Request Object/Parameters

StartHVHiPotTestRequest

- NetLists – {Collection<Connection>} – This defines the net lists you wish to test. Each net list is made up of a single "From" parameter, which should be a string value representing the test point range you wish to use as the "From Net" you wish to connect. The net list can use comma to delimit the test points, or hyphen to denote a range e.g. "1,2,3,5-9,12". You must specify multiple of these to make a complete short circuit test, the API will test shorts across each of these net lists to the other net lists defined
- VoltageV - {int} – The voltage to carry out the HV test at
- RampUpTimemS – {int} – The ramp up Time in milliseconds
- PreDwellTimemS – {int} – The time after ramp complete before measurement starts in milliseconds
- MeasureTimemS – {int} – The time in milliseconds to do the measurement for.
- RampDownTimemS – {int} – the time in seconds for the ramp down.
- UseAC – {bool} – A flag indicating whether to use AC or DC as the voltage and measurement source
- SwitchOffDevicesEMsBeforeandAfter – {bool} – Indicates whether the EMS and external devices will be switched off before and after the test. Defaults to true.
- Mode – {TestPointMode} – The mode which the test should be run in. See the TestPointMode section for possible values

4.4.2 Returns

A StartTestResult object.

4.5 RunExternalDeviceCommand – POST Method

Will run an action against an external device, you can find the available devices and their commands using the GetSupportedDevices method

4.5.1 Request Object/Parameters

RunExternalDeviceCommandRequest

- ExternalDeviceUniqueID - {string} – The Unique ID of the device to use
- CommandUniqueID - {string} – The Unique ID of the command to run
- MeasureTimems – {integer} – The time to measure for when using read action commands. This can be a maximum of 5 seconds (5000). The system will read from the device approx. every 20 milliseconds for this amount of time. If this is left at zero a single measurement will be taken.
- CommandParameters – {Collection<ExternalDeviceCommandParameter>} – A collection of the parameter and their value to set for the specified command. These can be retrieved from the GetSupportedDevices method call and each ExternalDeviceCommandParameter item should have the following properties:
    - ParameterID – {string} – The unique id of this parameter
    - ParameterValue – {string} – The value you wish to set this parameter to. This value must be within the valid range of values if one is set for this parameter

4.5.2 Returns

An RunExternalDeviceCommandResult object indicating whether the command was successful or not. This contains the following properties:

- Results – {Collection<string>} – This will only be populated for read action commands and will contain all the measurements taken during the MeasureTimems.
- ResultStatus – {ApiResultStatusType} – The status of the result
- ResultTime – {DateTime} – The time the result was generated
- LastUpdated – {DateTime} – The time the result was last updated, e.g. the time when the command started/completed
- ValidationFailures – {Collection<ValidationFailureItem>} – A list of validation failures, each item has the following properties:
    - ValidationResult – {ValidationFailureReason} – The reason for the validation failure
    - FailedItemValue – {string} – The value which failed validation

4.6 LoadtestProgram – POST Method

Loads a test program into the database

4.6.1 Request Object/Parameters

LoadTestProgramRequest

- Filepath- {string} – The path to the file to load

4.6.2 Returns

A LoadTestProgramResult object indicating whether the command was successful or not. This contains the following properties:

- Loaded – {bool} success or fail
- ID – {int} ID of test program if loaded else will be 0

4.7 DeleteAllTestPrograms – GET Method

Deletes all test programs from the database

4.7.1 Returns

An ApiResult object indicating whether the command was successful or not.

4.8 GetSupportedDevices – GET Method

Retrieves all the devices which can have an action run against them along with the commands and their parameters. This information can be used to build a RunExternalDeviceCommand request. This method has no additional parameters.

4.8.1 Returns

A GetSupportedDevicesResult with the following properties:

- ResultStatus – {ApiResultStatusType} – The status of the result
- ResultTime – {DateTime} – The time the result was generated
- LastUpdated – {DateTime} – The time the result was last updated, e.g. the time when the command started/completed
- Devices – {Collection<SupportedDevice>} – The devices supported in the system. Each item has the following properties:
    - ExternalDeviceUniqueID – {string} – The Unique ID for this device
    - ExternalDeviceName – {string} – The name of the device as set in the config
    - Commands – {Collection<SupportedDeviceCommand>} – The commands this device supports. Each one has the following properties:
        - CommandUniqueID – {string} - The unique ID of the command
        - CommandName – {string} – The user-friendly name of the command
        - IsReadActionCommand –{bool} – This indicates whether the command is a read action command or not. If it is a read action you will get results from running this command.
        - CommandParameters– {Collection<SupportedDeviceCommandParameter>} – The parameters for this command (if there are any). Each one has the following properties:

- ParameterID – {string} – Unique ID for this parameter
- ParameterName {string} – The user-friendly name of the parameter
- PossibleValues - {collection< SupportedDeviceCommandParamterPossibleValue>} – The possible values this parameter can have (if any). Each one has the following properties:
    - ValueID – {string} – The ID of the value – This is what would be set as the ParameterValue of the RunExternalDeviceCommandRequest
    - ValueName – {string} – The user-friendly name of the value

4.9 SwitchOffAllDevicesEMs – GET Method

Switch off all EMs and external devices which have a switch off command defined

4.9.1 Returns

An ApiResult with the following properties:

- ResultStatus – {ApiResultStatusType} – The status of the result
- ResultTime – {DateTime} – The time the result was generated
- LastUpdated – {DateTime} – The time the result was last updated, e.g. the time when the command started/completed

4.10 GetSystemInformation – GET Method

Retrieve information about the BLRT system and its components

4.10.1 Returns

A result object with the following properties:

- SerialNo - The system serial no
- SoftwareRevision - The revision number of the MK AT software
- ResultStatus – {ApiResultStatusType} – The status of the result
- ResultTime – {DateTime} – The time the result was generated
- LastUpdated – {DateTime} – The time the result was last updated, e.g. the time when the command started/completed

4.11 GetLatestResult – GET Method

Returns the last result from any of the Start Test methods. Optionally supports returning a specific test if the TestRequestID is supplied

4.11.1 Parameters

- TestRequestID - {int} - A specific test request ID, will instruct this method to return the result from that test request. If not supplied will return the last test result.
- IsRequestIDATestProgramID – {bool} – This defaults to false, if it's set to true, the response will be the latest result for the test program ID specified in the TestRequestID parameter. This allows you to retrieve the latest result of any test program, regardless of whether it was run via the API or not.

4.11.2 Returns

Depending on the test type, this will return one of the following result types:

- ContinuityTestResult
- ShortCircuitTestResult
- HVInsulationResult
- HVHiPotResult
- RunTestProgramResult

4.12 GetSubTestResult – GET Method

Returns a single sub test results from a test specified test

4.12.1 Parameters

- TestRequestID - {int} - A specific test request ID, will instruct this method to return the result from that test request. If not supplied will return the last test result.
- SubTestID – {int} – The Run ID of the required sub test.

4.12.2 Returns

A TestProgramResult object with just the selected subtest

4.13 StopRunningTest – GET Method

Stop the currently running test

4.13.1 Returns

A result object with the following properties:

- ResultStatus – {ApiResultStatusType} – The status of the result
- ResultTime – {DateTime} – The time the result was generated
- LastUpdated – {DateTime} – The time the result was last updated, e.g. the time when the command started/completed

4.14 GetTestStatus – GET Method

This will retrieve the estimated time left for a running test programmkengineer. When a test is being initialised, this will be the max value for a TimeSpan object which is 10675199.02:48:05.4775807 (10,675,199 days). On completion of a test the value will be 0

4.14.1 Returns

A GetTestStatusResult with the following properties:

- ResultStatus – {ApiResultStatusType} – The status of the result
- ResultTime – {DateTime} – The time the result was generated
- LastUpdated – {DateTime} – The time the result was last updated, e.g. the time when the command started/completed
- EstimatedTime – {TimeSpan} – Estimated time remaining for the test

4.15 GetAllTestPrograms – GET Method

This will retrieve a list of all test programs currently loaded into the runner.

4.15.1 Returns

A result object which is an array of objects with the following properties (or an empty array if no test programs loaded):

- ID – {int} – The unique ID used when referring to this test program
- PartNo/Name – {String} – The part number of the test program
- TestType – {int} – BLRT, Automeg
- Filename and path – {String} – The path and name of the file imported
- DateImported – {DateTimeOffset} – The time the test program was uploaded into the runner

Add the following for the completed message

- ResultStatus – {ApiResultStatusType} – The status of the result
- ResultTime – {DateTime} – The time the result was generated
- LastUpdated – {DateTime} – The time the result was last updated, e.g. the time when the command started/completed

4.16 StartTestProgram – POST Method

This will run the specified test program. There are some limitations to running a complete test via the API, more specifically:

- Operator instructions or inputs will not run and will cause a validation Failure with the validation result of “Invalid APITest Program” to be returned
- If an error occurs when communicating with an external device, you may get a message box shown on screen

- Although all possible options are supported in the test program, they may not be reported back through the API, or may be incomplete (e.g. AC offsets, HV shorts finding, continuity shorts finding, etc are excluded from the API results).

4.16.1 Request Object/Parameters

RunTestProgramRequest

- TestProgramID – {int} – The ID of the test program to run
- ReloadIfChanged– {bool} – reload test if file has changed from version in database.

4.16.2 Returns

A StartTestProgramResult object

5 Result Objects – Additional Information

5.1 StartTestProgramResult

This object is returned with any of the StartTest requests and consists of the following properties:

- TestRequestID – {int} – The unique ID for this request, which is required to be able to retrieve the result of the test. Will be -1 if the test failed to start. (*v10.9) Or this will be the ID of the test program if the GetLatestResult was called with the IsRequestIDATestProgramID parameter set to true.
- ResultStatus – {ApiResultStatusType} – The status of the result
- ResultTime – {DateTime} – The time the result was generated
- LastUpdated – {DateTime} – The time the result was last updated, e.g. the time when the test started/completed
- ValidationFailures – {Collection<ValidationFailureItem>} – A list of validation failures, each item has the following properties:
    - ValidationResult – {ValidationFailureReason} – The reason for the validation failure
    - FailedItemValue – {string} – The value which failed validation

5.2 StartTestResult

This object is returned with any of the StartTest requests and consists of the following properties:

- TestRequestID – {int} – The unique ID for this request, which is required to be able to retrieve the result of the test. Will be -1 if the test failed to start.
- Type – {SubTestTypes} – The type of the test.
- ResultStatus – {ApiResultStatusType} – The status of the result
- ResultTime – {DateTime} – The time the result was generated
- LastUpdated – {DateTime} – The time the result was last updated, e.g. the time when the test started/completed
- ValidationFailures – {Collection<ValidationFailureItem>} – A list of validation failures, each item has the following properties:
    - ValidationResult – {ValidationFailureReason} – The reason for the validation failure
    - FailedItemValue – {string} – The value which failed validation

5.3 ContinuityTestResult

This result will contain all the properties of the StartTestResult, along with the following extra properties:

- Connections – {Collection<ConnectionResult>} – The result of each connection being tested. There will be 1 entry for each of the connection supplied in the StartContinuityTestRequest and each entry will have the following properties:

5.3.1 ConnectionResult

- From – {string} – The from test point
- To – {string} – The to test point
- MeasuredResistanceOhms – {double} – The measured resistance of the connection in Ohms
- AppliedCurrentmA – {double} – The current which was applied to read the resistance in milliamps
- TestedStateDetail – {TestedStateSubType} - The result of this measurement

5.4 ShortCircuitTestResult

This result will contain all the properties of the StartTestResult, along with the following extra properties:

- ShortedNetLists – {Collection<ShortCircuitNetListResult>} – The list of nets which were shorted, this collection will be empty if there were no shorts found. Each entry will have the following properties:

5.4.1 ShortCircuitNetListResult

- From – {string} – The net list which was shorted from. There may be multiple entries in the ShortedNetLists collection with the same from net if the from was shorted to multiple points
- To – {string} – The net list which was shorted to
- MeasuredResistanceOhms – {double} – The measured resistance of the connection in Ohms
- AppliedCurrentmA – {double} – The current which was applied to read the resistance in milliamps

5.5 HVHiPotResult

This result will contain all the properties of the StartTestResult, along with the following extra properties:

- NetListResults– {Collection<HVHiPotNetListResult>} – The result of each net list being tested. There will be 1 entry for each of the connections supplied in the StartHVHiPotTestRequest and each entry will have the following properties:

5.5.1 HVHiPotNetListResult

- From – {string} – The net list which was tested
- MeasurementState – {HVMeasurementState} – The result of this HV measurement – whether there was a flash or surge or whether it was successful

- AverageMeasuredCurrentmA – {string} – The average current which was measured over the measurement time
- LowestMeasuredCurrentmA – {string} – The lowest current which was measured over the measurement time
- HighestMeasuredCurrentmA – {string} – The highest current which was measured over the measurement time

5.6 HVInsulationResult

This result will contain all the properties of the StartTestResult, along with the following extra properties:

- NetListResults– {Collection<HVInsulationNetListResult>} – The result of each net list being tested. There will be 1 entry for each of the connections supplied in the StartHVInsulationTestRequest and each entry will have the following properties:

5.6.1 HVInsulationNetListResult

- From – {string} – The net list which was tested
- MeasurementState – {HVMeasurementState} – The result of this HV measurement – whether there was a flash or surge or whether it was successful
- AverageMeasuredResistanceMOhms – {string} – The average resistance which was measured over the measurement time
- LowestMeasuredResistanceMOhms – {string} – The lowest resistance which was measured over the measurement time
- HighestMeasuredResistanceMOhms – {string} – The highest resistance which was measured over the measurement time

5.7 StartTestProgrameResult

This result will contain all the properties of the StartTestResult, along with the following extra properties:

- TestedState – {TestedState} – The current state of the overall test program.
- StartedDate – {DateTime} – The date/time that the test started
- CompletedDate – {DateTime} – The date/time that the test completed
- SubTests – {Collection<SubTestResult>} – There will be an item for each sub test in the test program. Each entry will have the following properties:

5.7.1 SubTestResult

- TestedState – {TestedState} – The current state of this sub-test. You can track progress of a test by looking at the number of untested sub tests
- TestType – {SubTestTypes} – The type of this sub test

- ShortCicuitParameters– {Collection<ShortCircuitNetListResult>} – Results of short circuit sub test
- HiPotParameters– {Collection< HVHiPotNetListResult>} – Results of short circuit sub test
- ContinuityParameters– {Collection< ConnectionResult>} – Results of short circuit sub test
- InsulationParameters– {Collection< HVInsulationNetListResult>} – Results of short circuit sub test
- AdvanceParameters– {Collection< AdvanceResult>} – Results of short circuit sub test
-

5.7.2 AdvancedResult

- From – {string} – The net list which was tested
- To – {string} – The net list which was tested
- ExternalDeviceUniqueID – {string} – The Unique ID of the device being measured
- CommandName– {string} – The name of the command of the device being measured
- AverageMeasured – {string} – The average which was measured over the measurement time, the measurement unit will depend on the device you are reading from
- LowestMeasured – {string} – The lowest which was measured over the measurement time, the measurement unit will depend on the device you are reading from
- HighestMeasured – {string} – The highest which was measured over the measurement time, the measurement unit will depend on the device you are reading from

6 Real-Time Updates

The MK AT also hosts a SignalR server which will publish status updates and results of tests that have been started. You can subscribe to the following events:

6.1 StatusChanged

Will report a status change, the object it contains has the following properties:

- ServerStatus - The new status - see the Server Status section for more information
- Message – A user- friendly message of the status change

6.2 TestCompleted

Will publish each time a test is completed and returns the same object as a GetLatestResult request

7 Expected Values

This section lists the possible values which a specific property type can contain, see the above method request/result objects to find their use

TestPointMode
TwoWire
FourWire

SubTestTypes
Continuity
ShortCircuit
Insulation
ACHiPot
DCHiPot
TestProgram

ApiResultStatusType
TestInProgress
Error_SubTestAlreadyRunning
Error_Timeout
Error_GeneralFailure
Error_IncorrectProbes
Error_InternalSystemErrror
Error_ResistanceTooHigh

Error_UserAborted
Error_ValidationFailure

ValidationFailureReason
RequestBodyIsNotCorrectStructure
ItemBeingValidatedIsNotSet
TestPointInvalid
TestPointOutsideRange
TestPointMasked
CurrentBelowMinimum
CurrentAboveMaximum
VoltageAboveMaximum
DwellBelowMinimum
DwellAboveMaximum
ResistanceLimitBelowMinimum
ResistanceLimitAboveMaximum
NoConnectionsSpecified
RampUpTimeBelowMinimum
RampUpTimeAboveMaximum
PreDwellTimeBelowMinimum
PreDwellTimeAboveMaximum
MeasureTimeBelowMinimum
MeasureTimeAboveMaximum
RampDownTimeBelowMinimum

RampDownTimeAboveMaximum
ExternalDeviceTriggerContainsNoSequence
DeviceNotFoundInConfiguration
CommandNotFoundInConfiguration
CommandParameterNotFoundInConfiguration
CommandParameterValueBelowMinimum
CommandParameterValueAboveMaximum
CommandParameterValueNotValidEnumEntry
CommandParameterValueNotWithinListOfAvailableOptions
ExternalDeviceNotFound
ExternalDeviceCommandNotFound
CommandParameterNotRecognised
ExternalDeviceUniqueIDInvalid
CommandUniqueIDInvalid
TestPointModeNotSet
SubTestIsNotSet
VoltageBelowMinimum

TestedState
Untested
Passed
Failed
Aborted
Error

Partial
Running

TestedStateSubType
ResistanceTooLow
ResistanceOverLimit
ValidMeasurement

HVMeasurementState
FlashDuringMeasure
NotReadyAfterRamp
SurgeAtTopOfGain1
MeasureBelowThreshold
MeasureAboveThreshold
ValidMeasurement
SystemTurnedOff
NoMeasurements
SurgeMovingFromGain10ToGain1
NotReadyDuringMeasure
FlashDuringRampUp
FlashDuringRampDown
FlashDuringPreDwell

ServerStatus
ErrorOccurred
ContinuityTestStarted
ShortCircuitTestStarted
HVHiPotTestStarted
HVInsulationTestStarted
GettingSystemInformation
GettingSystemInformationCompleted
GetLatestResultRequested
GetLatestResultCompleted
StopTestRequested
StopTestCompleted
TestCompleted
Stopping
Started
ExternalDeviceCommandStarted
ExternalDeviceCommandCompleted
GetSupportedDevicesStarted
GetSupportedDevicesCompleted
SwitchOffAllDevicesEMsStarted
SwitchOffAllDevicesEMsCompleted

8 Examples

8.1 Via Javascript/SignalR

8.1.1 Important Notes

Due to security restrictions these calls may fail when run in Microsoft Edge unless the web page is hosted within an internet context on the same host (e.g. IIS).

Other browsers should work OK even when using a html file on the filesystem.

For an example of these calls you can use the built in MK AT API Control Panel (see Testing the API above).

8.1.2 Example 1: To start a continuity test returning JSON:

Note: You will need a reference to the jQuery script file in your html

```
$.ajax({
    url: "http://localhost:8080/StartContinuityTest",
    crossDomain :true,
    cache: false,
    data: '{"Connections":[{"From":1,"To":3},{"From":6,"To":11}],'+
            '"VoltageLimitV" : 30, '+
            '"CurrentmA" : 50, '+
            '"DwellmS" : 10, '+
            '"MaxMeasuredResistanceOhms" : 1,'+
            '"Mode" : "TwoWire"}',
    type: "POST",
    headers: { 'Cache-Control': 'no-cache' , 'Content-Type':
'application/json', 'Accept': 'application/json'},
    success: function(json){
        alert(json.ResultStatus);
     },
    error : function(jqXHR, textStatus, ex){
        alert("Error has occurred " + textStatus + ", " + ex);
    }});
```

8.1.3 Example 2: To start a continuity test returning XML:

Note: You will need a reference to the jQuery script file in your html

```
$.ajax({
    url: "http://localhost:8080/StartContinuityTest",
    crossDomain :true,
    cache: false,
    data: "<StartContinuityTestRequest> " +
           "<Connections> " +
           "<Connection><From>1</From><To>3</To></Connection> " +
           "<Connection><From>6</From><To>9</To></Connection> " +
           "</Connections> " +
           "<VoltageLimitV>30</VoltageLimitV> " +
           "<CurrentmA>50</CurrentmA> " +
           "<DwellmS>10</DwellmS> " +
           "<MaxMeasuredResistanceOhms>1</MaxMeasuredResistanceOhms> " +
           "<Mode>TwoWire</Mode> " +
           "</StartContinuityTestRequest>",
    type: "POST",
    headers: { 'Cache-Control': 'no-cache' , 'Content-Type':
'application/xml', 'Accept': 'application/xml'},
    success: function(xml){
        alert((new XMLSerializer()).serializeToString(xml));
     },
    error : function(jqXHR, textStatus, ex){
        alert("Error has occurred " + textStatus + ", " + ex);
    }});
```

8.1.4 Example 3: To listen for status updates:

Note: You will need a reference to the jQuery script file and the ASP.NET SignalR JavaScript Library in your html

```
var hubConnection = $.hubConnection("http://localhost:8080/signalr");
var hub = hubConnection.createHubProxy("mKATSignalRHub");
if (hub  === undefined)
{
    alert ("Error connecting to SignalR hub");
}
else
{
    hub.on("StatusChanged", UpdateServerStatus);
    hubConnection.start();
}

function UpdateServerStatus(newstatus, message) {
    alert(message);
}
```

8.1.5 Example 3: To retrieve the result ID 1001:

Note: You will need a reference to the jQuery script file in your html

```
$.ajax({
   url: "http://localhost:8080/GetLatestResult/1001",
   crossDomain :true,
   cache: false,
   type: "GET",
   headers: { 'Cache-Control': 'no-cache' , 'Accept':'application/json'},
   success: function(json){
       alert(json.ResultStatus);
    },
   error : function(jqXHR, textStatus, ex){
       alert("Error has occurred " + textStatus + ", " + ex);
   }});
```

8.2 Via C#

8.2.1 Example 1: To start a continuity test:

Note: To convert the results from a JSon string to an object we will use the Newtonsoft.Json NuGet package

```
var url = "http://localhost:8080/startcontinuitytest";
WebRequest req = WebRequest.Create(url);

req.ContentType = "application/json";
req.Method = "POST";

byte[] bytes = Encoding.ASCII.GetBytes("{'Connections':[{'From':1,'To':3}," +
"{'From':6,'To':11}],"+
"'VoltageLimitV' : 30, " +
"'CurrentmA' : 50, " +
"'DwellmS' : 10, " +
"'MaxMeasuredResistanceOhms' : 1," +
"'Mode' : 'TwoWire'}");
req.ContentLength = bytes.Length;

using (Stream os = req.GetRequestStream())
{
    os.Write(bytes, 0, bytes.Length);
}
var result = string.Empty;
using (WebResponse resp = req.GetResponse())
{
    if (resp != null)
    {

        using (StreamReader sr = new StreamReader(resp.GetResponseStream()))
        {
            result = sr.ReadToEnd().Trim();
        }
    }
}

if (!string.IsNullOrEmpty(result))
{
    var json = JObject.Parse(result);
    Console.WriteLine(json["ResultStatus"]);
}
```

8.2.2 Example 2: To listen for status updates:

Note: You will need to install the Microsoft.AspNet.SignalR.Client NuGet package

```
var connection = new HubConnection("http://localhost:8080/signalr");
var hubProxy = connection.CreateHubProxy("mKATSignalRHub");
hubProxy.On("StatusChanged", (string status, string message) =>
{
      Console.WriteLine("Incoming data: {0}, {1}", status, message);
});
hubProxy.On("TestCompleted", (JObject result) =>
{
      Console.WriteLine("Incoming data: {0}", result);
});
ServicePointManager.DefaultConnectionLimit = 10;
await connection.Start();
```