TrueTest MPK API Specification		
Version:	1.0.2	
Subject:	*MPK API specifications for TrueTest version 1.7. or higher.*	Research and Development Application Note
Contents:	14 Pages, No Disks	October 2019

Application Note History

Application Note Name: Park TrueTest CLI API.docx
Title: Park TrueTest CLI API Specifications
Product: TT_API.dll
Versions: MPK_API 1.0.6262.17976 and higher &
TT_API 1.0.6194.20809 and higher
Original Author: EMattson
Creation Date: January 2017
Release Date: February 2017

Updates

Revision Date	Changes by/ Comments
2017-02-22	Added 'useLogging' argument to CloseCommunicationAndReinitializeCamera Method to match Initialize Method. Added ability for API to use WhiteListKey file.
2017-03-02	Updated JSON object key values so user knows how to parse.
2017-05-22	Updated GetSerialNumber method and Initialization *showFeedbackUI* parameter description.
2017-07-13	Updated CreateMeasurementSetup method to use additional arguments.
2017-08-17	Added GetLastMeshData and GetRawData methods to API
2019-10-03	New MPK API Documentation – Evan Atchison

Radiant Vision Systems, 22908 NE Alder Crest Dr. Ste 100
+1 425 844 0152 Voice • +1 425 844 0153 Fax

Email : support@radiantvs.com
Internet: http://www.radiantvs.com

Table of Contents

MPK_API Class - General

Background

General functions to initialize the camera and retrieve the camera serial number.

Public Methods

InitializeCamera Method

Initializes camera and components of TrueTest.

Syntax

Declaration

public string InitializeCamera(string cameraSerial, bool showFeedbackUI, bool useLogging)

Parameters

cameraSerial

The camera serial string to initialize. If the serial is "Demo," then use demo camera.

showFeedbackUI

If true, show windows and forms while initializing API / during take measurement. If false this will hide any camera status forms and will also hide lower level message boxes that may appear.

useLogging

If true, write to debug log while initializing API.

Returns

A JSON format string: {"ErrorCode": "Success = 0 | Unknown = 1 | InitializationFailure = 2"}

GetCameraSerial Method

Returns the attached camera's serial number.

Syntax

Declaration Public Function

GetCameraSerial() As JObject

Parameters

None

Returns

A JSON object. If no camera is attached or powered on, JSON value = "NONE." If exception occurred, return is JSONUnknownException. {"CameraSerial": "NONE | *CameraSerialString*"} If return value is "NONE" then the user should NOT proceed with the camera initialization; otherwise, the camera discovery dialog window will appear.

EquipmentReady Method

Returns whether camera has initialized or not.

Syntax

Declaration Public Function

EquipmentReady() As JObject

Parameters

None

Returns A JSON object: {"ErrorCode": "Success = 0 | Unknown = 1 | InitializationFailure = 2"}

CloseCommunication Method

Shuts down API objects and communication to camera.

Syntax

Declaration Public Function CloseCommunication() As JObject

Parameters

None

Returns A JSON object: {"ErrorCode": "Success = 0 | Unknown = 1 | InitializationFailure = 2"}

CloseCommunicationAndReinitializeCamera Method

Shuts down API objects and communication to camera, then reinitializes connection and communication.

Syntax

Declaration

Public Function CloseCommunicationAndReinitializeCamera(cameraSerial as String, showFeedbackUI As Boolean, useLogging as Boolean) As JObject

Parameters

cameraSerial

The camera serial string to initialize.

showFeedbackUI

If true, show windows and forms while initializing API / during take measurement.

useLogging

If true, write to debug log while initializing API.

Returns

A JSON object: {"ErrorCode": "Success = 0 | Unknown = 1 | InitializationFailure = 2"}

GetTrueTestApiVersionInfo Method

Returns current version of TrueTestEngine DLL.

Syntax

Declaration Public Function GetTrueTestApiVersionInfo As JObject

Parameters

none

Returns

A JSON object: {"TrueTestApiVersioninfo":"*VersionInfo"}* or {"ErrorCode": Unknown = 1 | InitializationFailure = 2"}. If success, version returns as File Description + Product Version (major revision.minor version.build number.revision number).

GetMpkApiVersionInfo Method

Returns current version of MPK_API DLL.

Syntax

Declaration Public Function GetMpkApiVersionInfo As JObject

Parameters

none

Returns

A JSON object: {"MpkApiVersionInfo":"*VersionInfo"}* or {"ErrorCode": Unknown = 1 | InitializationFailure = 2"}. If success, version returns as: File Description + Product Version (major revision.minor version.build number.revision number).

Sequence

Functions related to preparing and running a pre-defined TrueTest sequence.

Public Methods

SetSequence Method

Sets the sequence file .

Syntax

Declaration

```
public string SetSequence(string sequenceFilePath)
```

Parameters

sequenceFilePath

Full file path of the target sequence file.

Returns

JSON formatted string Success or error message.

RunAllSequenceSteps

Returns the last sequence run results as a list of string.

Syntax

Declaration

```
public string RunAllSequenceSteps(bool useCamera = true, bool saveImages = true)
```

Parameters

useCamera (Optional)

Whether to use the camera to take a new image for the sequence or to use measurements currently in the database. Defaults to true. If the software is in Demo mode without a camera, this will always be false.

saveImages(Optional)

Whether to save the camera measurements to a database. If useCamera is set to false, will always be set to false.

Returns

JSON Serialized string that deserializes to a Dictionary of List of Dictionaries containing analysis results.

The top-level dictionary is keyed by analysis name, and then has a value that is a list of results reported for that analysis.

Each result in the result list is a dictionary that has a key which is the result property name and the result property value. The available result property names are:

- *Analysis ID:* Identifying number for the analysis generating the result
- *Analysis Name:* Username of the analysis used
- *DateTime:* Time the result was generated
- *Name:* Name of the result
- *PassFail:* If the result has a Pass/Fail criteria, outputs whether it passes or fails
- *PatternName:* Name of the Pattern the Analysis ran on.

- *SequenceName:* Name of the Sequence the Analysis is contained in
- *SerialNumber:* Serial Number of the Measurement the Analysis ran on
- *Unit:* Unit of the result, if applicable
- *Value:* Value of the result, if applicable
- *ValueString:* String value of the result, if applicable

Example return:


```
{'Particle Defects': [
        {
            'AnalysisID': 22,
            'AnalysisName': 'Particle Defects',
            'DateTime': '2012-02-16T14:54:43',
            'Name': 'NumDarkParticles',
            'PassFail': 1,
            'PatternName': 'White',
            'SequenceName': 'ConoscopeTest',
            'SerialID': 8,
            'SerialNumber': 'Demo',
            'Value': 0.0,
            'ValueString': '0'
        },
        {
            'AnalysisID': 22,
            'AnalysisName': 'Particle Defects',
            'DateTime': '2012-02-16T14:54:43',
            'Name': 'NumDarkBlobs',
            'PassFail': 1,
            'PatternName': 'White',
            'SequenceName': 'ConoscopeTest',
            'SerialID': 8,
            'SerialNumber': 'Demo',
            'Value': 0.0,
            'ValueString': '0'
        },
```


RunSequenceStepByName

Returns the last sequence run results as a list of string.

Syntax

Declaration

```
public string RunSequenceStepByName(string stepName,bool useCamera = true, bool
        saveImages = true)
```

Parameters

stepName

Username of the analysis to run.

useCamera (Optional)
Whether to use the camera to take a new image for the sequence or to use measurements currently in the database. Defaults to true. If the software is in Demo mode without a camera, this will always be false.

saveImages(Optional)
Whether to save the camera measurements to a database. If useCamera is set to false, will always be set to false.

Returns
Same return as RunAllSequenceSteps.

RunSequenceStepListByName
Returns the last sequence run results as a list of string.

Syntax
Declaration
```
public string RunSequenceStepByName(List<string> stepNames, bool useCamera = true, bool
       saveImages = true)
```

Parameters
stepName
Username of the analysis to run.

useCamera (Optional)
Whether to use the camera to take a new image for the sequence or to use measurements currently in the database. Defaults to true. If the software is in Demo mode without a camera, this will always be false.

saveImages(Optional)
Whether to save the camera measurements to a database. If useCamera is set to false, will always be set to false.

Returns
Same return as RunAllSequenceSteps.

Data Export

Functions to export measurement data to disk.

Public Methods

ExportMeasurement Method

Exports the luminance and color data to a file for the given image key.

Syntax

Declaration

```
public string ExportMeasurement(string imageHandle, string path, string fileName = "",
int resolutionX = 0, int resolutionY = 0)
```

Parameters

imageHandle

The name or database ID of the measurement that was captured.

path

The folder location where the exported data will be written.

filename (Optional)

The name of the export file. If the name includes one of the extensions {.csv, .png, .jpg, .bmp} that extension will be used. Otherwise, the export will default to .csv. If no *filename* is passed in, a default name will be created and exported as .csv.

resolutionX (Optional)

Resolution of export in the X direction for .csv export. If none, full size image will be exported (this takes a long time).

resolutionY (Optional)

Resolution of export in the Y direction for .csv export. If none, full size image will be exported (this takes a long time).

Returns

A JSON string with the filepath of the exported measurement or an error message.

Measurement Database

Functions related to setting calibrations and creating a measurement setup used during image capture.

Public Methods

SetMeasurementDatabase Method

Sets which measurementDatabase to use.

Syntax

Declaration

```
public string SetMeasurementDatabase(string databasePath)
```

Parameters

databasePath

The full file path to the database

Returns

Json string of the database path or error message.

CreateMeasurementDatabase Method

Creates a new measurementDatabase to use.

Syntax

Declaration

```
public string CreateMeasurementDatabase(string databasePath)
```

Parameters

databasePath

The full file path to the database

Returns

Json string of the database path or error message (will throw error if database already exists).

GetMeasurementList

Gets a list of measurements and some relevant information.

Syntax

Declaration

```
public string GetMeasurementList()
```

Parameters

None

Returns

A JSON string that deserializes to a List of Dictionaries. Each Dictionary in the List corresponds to one measurement in the database. Each Dictionary has four keys:

- *Description:* Measurement name.

- *Measurement ID:* Unique ID of the measurement.
- *Measurement Setup:* Name of the measurement setup.
- *Pattern:* Name of the measurement pattern.

GetMeasurementInfo Method

Returns the color calibrations.

Syntax

Declaration

```
public string GetMeasurementInfo(string measurementName)
```

Parameters

measurementName

Name or Database ID of a measurement to look for.

Returns

JSON Serialized Dictionary containing many properties of the specified measurement.

EditMeasurementInfo Method

Returns the image scale calibrations.

Syntax

Declaration

```
public string EditMeasurementInfo(string measurementName, string JsonMeasInfo)
```

Parameters

measurementName

Name or Database ID of a measurement to look for.

JsonMeasInfo

JSON formatted string that must deserialize into a Dictionary<string, string>. The available dictionary keys are:

- "Description" : the serial number of the measurement
- "Model Number" : the sequence used to take the measurement
- "Technician" : the Analysis used to take the measurement
- "Pattern" : Pattern Name
- "Measurement Setup" : Measurement setup name

The value of each key will be set to the corresponding property of the designated

Returns

A JSON serialized string of a success or error message.

MPK_API Class – Measurement

Background

Functions related to capturing measurements.

Public Methods

CaptureMeasurementWithPattern Method

Captures a measurement using a pre-defined pattern setup and stores it in the measurement database.

Syntax

Declaration

```
public string CaptureMeasurementWithPattern(string patternSetupName, string
measurementName)
```

Parameters

patternSetupName

The name of the pattern setup.

measurementName

The name of the measurement in the database.

Returns

A JSON string indicating if the measurement capture succeeded or failed.

GetListOfPatternSetups Method

Get the current image key names stored in memory.

Syntax

Declaration

```
public string GetListOfPatternSetups()
```

Parameters

None

Returns

JSON string that deserializes to a list of strings of the pattern setup names.

Appendix I

White List Key file

The ability to use a white list key file – to allow TrueTest to run without a dongle attached – has been added for MPK_API version > 1.0.6262.17976 and TrueTestEngine version > 1.2.0.1136.
The WhiteListSecurity.xml file must be in the \Radiant Vision Systems Data\TrueTest\AppData folder.
As of this current writing, February 23, 2017, Radiant Imaging Colorimeters and Photometers still require the entry of the PMEngine license code *if and only if*:

1. The PMEngine license code hasn't been previously entered for a particular camera purchased before 2017/02/23.
2. If the camera calibration database (.calx) for a camera, that was used on a system and that contains the entered PMEngine license code, is not moved to the new system where the camera is being used.
    a. If the calibration file is new, meaning it was copied from the camera, and the camera was received before the date listed above, a message box will appear requiring the user to enter the PMEngine license code.
    b. To avoid the PMEngine license code pop-up, the user shall copy the previously used calx camera calibration file to the new machine, or use a local (network) location that houses all calx camera calibration files.

For cameras purchased *after* the listed date above: the user will not be required to enter a PMEngine license code.