GSM Async API (external) 3.7.103

4/19/2024

Objective

Describe the GelSight Mobile (GSM) Async API for external software developers.

Use Cases

The API is developed for two main use cases:
- Automation
- Extension of manual use of GSM

Automation

Enable the integration of GSM (GelSight Mobile) into a fully automated workflow where GSM and a GelSight sensor are controlled components in a robotic system. The intended scenario is in-line inspection in a manufacturing setup. Possible other components in the system could be a robot holding the GelSight sensor and maybe another robot that controls a conveyor belt where manufactured parts are inspected and then pushed forward.

From a system perspective, GSM and the sensor are considered *components*, and not the entity in charge of the manufacturing operation. The Async API is designed with that mindset.

Given that most operations in GSM are orders of magnitude longer than control loop operations like moving a robot and monitoring the move, the chosen service provider methodology is that of an event-based system, where the controlling entity subscribes to relevant events and subsequently orders GSM to start working on completing different tasks, where the success/failure is reported back as events. This methodology has the added benefit of allowing human interaction with the workflow, e.g. allowing a human to press the scan button and then use the event raised at completion of a scan to trigger the next operations managed by the controlling entity.

Extension of manual use of GSM

The GSM Async API will raise events to an external app each time an analysis method is completed and saved by GSM. This will be a rich event that contains all outputs and inputs for this analysis method. Reporting this data to an external app enables the application developer to develop custom logic that incorporates the results of multiple scans. It also allows the program logic to take input from other systems such as an ERP/SAP system or other customized workflow software used by the customer.

Some customers may have a desire to automatically forward results generated by GSM into their existing data systems. Prior to the API, this was only possible by manually transferring results from GSM into the customers own system. By writing a simple external app that integrates the GelSight Async DLL, the app will receive events with the results of any analysis. The developer can then easily decide which section of the outputs will be transferred to their own data system, either automatically or via customized UI.

Some customers need to trigger the scan based on their own workflow but perform the analysis inside GSM. This could be using a hardware foot pedal to trigger a scan or maybe some other software system that needs to trigger a scan. Using the API, a single line of code can be written to trigger a scan. It is even possible to instruct GSM what to name the scan and where to store it, e.g. on a shared network drive.

Environment

The GelSight Async API is delivered as a .NET DLL. The DLL can be linked into multiple programming languages as long as .NET8 is installed on the computer.

The operating system supported is Microsoft Windows 11. Microsoft Windows 10 is supported until October 14, 2025.

The API requires a licensed GelSight Mobile to be installed.

The API communicates with GelSight Mobile via a TCP socket connection. This means the application developer has the choice of running the custom app with the DLL on the same computer as GSM or on a networked computer.

Getting Started

To enable API developers a quick and easy start, an open-source Visual Studio sample project has been developed with a "Hello World" application. The Hello World application is only about 300 lines of code that demonstrate all key functions of the API.
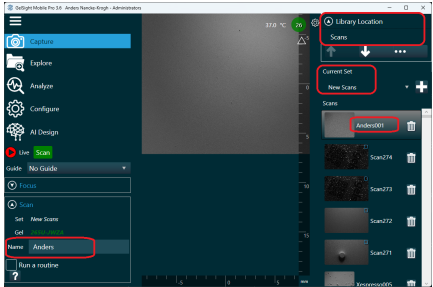
The sample is written in C#, but using simple structures so that non-C# developers can easily test the behavior and follow the logic.

Assuming the developer has installed and licensed Gelsight Mobile as well as installed Visual Studio, the project will compile and run in a few minutes.

Actions

The following actions can be requested by the application developer via the DLL. Details of the parameters and data structures are documented in the sample code.

NOTE: Value types with a '?' are nullable in C#.

Name	Properties	Description
RequestScan	• ScanPath [string?] • ScanPrefixName [string?]	In GSM, the scan folder is created based on the following properties managed via the GSM GUI. • Library Location (in this example "Scans") • Current Set (in this example "New Scans") • Name (in this example "Anders") The scan folder is created as a subfolder in <Library Location>/<Current Set>. The scan folder is named <Name> <enumerator>. The parameters of this method are, • ScanPath, which is used to replace <Library Location>/<Current Set>. • ScanPrefixName, which is used to replace the name field in the lower left corner of the GSM user interface. The final name for the scan will be generated by GSM based on enumeration of the existing scan names in order to not create duplicate scan folders.

		If either the ScanPath or ScanPrefixName string is an empty string or null, then the scan will be saved using the current UI settings described above.
RequestAnalysis	• ScanFolder [string] • AnalysisName [string]	The ScanFolder parameter is the path to the scan data to be used by the analysis. The name of the analysis to run on the scan, is defined by the AnalysisName parameter. If the analysis name is incorrect or refers to an unlicensed analysis type, then the ErrorMessageReceived event will be raised by the ConnectionManager. This setup will support variants as well, so a GSM admin can create a variant of a baseline analysis method, modify the parameters, e.g. Pass/Fail to their needs and then make it available to the API. The client simply needs to pass the name of the variant analysis in the AnalysisName parameter. After running the analysis, the result is automatically saved and the ConnectionManager will raise the AnalysisSaved event. Critical note If the scan is already displayed in the "Analyze" view in GSM, any changes made by the user will be saved without notifying the user.
RequestLiveView	• Active [bool?]	This request will only change the live view status if a device is

|		currently connected, and it is possible to navigate to the Capture view. Setting the Active parameter to false will turn the camera off to save power. Setting the Active parameter to true will navigate GSM to the Capture view and turn the camera on. Setting the Active parameter to null will have the same effect as setting it to true. NOTE: GSM can never guarantee a change to live view as multiple conditions could prohibit this from taking place, for example: - A scan is left open with changes in GSM. The user would need to save before a change can be made. - The sensor is disconnected. - The gel is not calibrated. - Other system error conditions like out of disk space, etc.
TryDeleteScan	• ScanFolder [string]	GSM will attempt to send the folder passed in by the ScanFolder parameter to the Windows Recycle bin. The ScanDeleted event will then be raised with the result.
RequestHeightmap	• ScanFolder [string] • Align [bool?] • Detrend [bool?] • Replica [bool?] • Detrend Order [int?] (1-4)	GSM refers to the 3D model of the surface as a "heightmap". On disk this data is saved in a tmd file, which is a binary file containing the Z height value of each data point on the surface. GSM will raise the HeightmapStarted and the

HeightmapCompleted events to notify the client the status of the height map creation.

Setting the Align parameter to true will run the 'Auto-Align Images' algorithm when creating the heightmap. Setting the Align parameter to false will not run the 'Auto-Align Images' algorithm when creating the heightmap.

Setting the Detrend parameter to true will run the 'Form Removal' algorithm when creating the heightmap. Setting the Detrend parameter to false will not run the 'Form Removal' algorithm when creating the heightmap.

Setting the Replica parameter to true will run the 'Replica Transform' algorithm when creating the heightmap. Setting the Replica parameter to false will not run the 'Replica Transform' algorithm when creating the heightmap.

Setting the Replica, Detrend, or Replica parameter to null will honor the GSM setting for this option as shown in the image below.
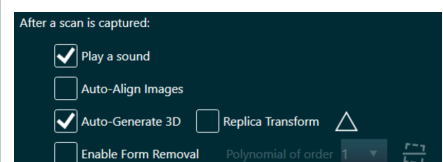
Events

The Client DLL's ConnectionManager will raise the following events when state changes in GSM. For more information, please refer to the Hello World sample project on GitHub.

Name	Properties	Description
ConnectionStateChanged	ConnectionState	This event is raised when the GSM connection state has changed. The possible connection states are: • Not Started • Connecting • Connected • Connected And Validated • Disposing • Validation Failed • Version Mismatch
StatusChanged	• IsSensorPresent [bool] • IsSensorLive [bool] • IsScanPossible [bool] • Model [string?] • Serial [string?] • Configuration [string?] • FirmwareVersion [string?] • NumberOfChannels [int?] • ImageWidth [int?] • ImageHeight [int?]	This event is raised when, • A sensor is connected or disconnected. • Entering or leaving the GSM live view. • The sensor ability to make a capture has changed. Note that some of the properties can be null if their data is not defined. For example, the Serial parameter is an empty string when IsSensorPresent is false.
GelChanged	• Serial [string?] • Type [string?] • UseBy [DateTime?]	The event is raised when a gel is changed. Notice that only the Serial property is guaranteed to have a valid string.
ScanCompleted	• ScanFolder [string] • ScanMetaData [string]	The event is raised when a scan is completed and all files have been saved to the scan folder. The ScanFolderparameter is the folder where GSM has saved all files related to the scan.

		The ScanMetaData parameter is the content of the yaml file formatted as a JSON string. Refer to the ScanCompleted Event JSON section later in this document for more information.
HeightmapStarted	• ScanFolder [string]	The event is raised when 3D generation of scan data is started.
HeightmapCompleted	• ScanFolder [string] • Success [bool]	The event is raised when 3D generation of scan data is completed and has been saved to the file system.
AnalysisSaved	• ScanFolder [string] • Success [bool] • Results [string] (json formatted) • ErrorMsg [string] • RequestId [int]	The event is raised when an analysis of a scan has been <u>saved</u> to the file system. The ScanFolder parameter is the folder where GSM saved all files related to the scan. The Success parameter indicates if the requested analysis was successful. The Results parameter is the contents of the scancontext.yaml file formatted as a JSON string. It contains the user created shapes as well as user inputs and generated outputs of all analysis methods run on this scan. The API caller will be able to retrieve PASS/FAIL information directly from the Results JSON string. The ErrorMsg parameter contains a string with information regarding why an analysis failed to run. The RequestId is the id of the client message used to initiate the analysis. It may be 0 if the analysis was not initiated remotely.
ScanDeleted	• ScanFolder [string] • Success [bool]	This event is raised after the TryDeleteScan request is made or if the

		user deletes a scan in GSM. The Success parameter represents if the deletion operation succeeded or failed. NOTE: If a scan is deleted outside of GSM, such as in Windows Explorer for example, the API will not raise this event.
ErrorMessageReceived	• ErrorMessage [string] • RequestId [int]	This event is raised if an error occurs in GSM while, • Capturing a scan. • Creating a heightmap. • Running an analysis method. • A hardware error has occurred. The RequestId is the id of the client action that was being processed. It may be 0 if the operation was not initiated by the Client program.

ScanCompleted Event

The ScanMetaData parameter in the ScanCompleted event contains JSON formatted data from the scan.yaml file located in the newly captured scan's folder on disk. It is recommended to use the free C# Newtonsoft Json.NET library to help parse this information, but the API developer is free to use any method that they choose.

The JSON string does contain some replication of data.

Header	Example	Comment
version	2.0	Version of the json structure
guid	dd773e2f-e0f7-4dbc-94c9-350b7de4e28c	Unique Id
createdon	2023-07-19 10:42:21	Date/time of scan capture
mmperpixel	0.0069981396828	Resolution of heightmap and images [mm per pixel]
sdkversion	3.8.0	Version of

		internal GSM math library used in analysis methods
crop	(0, 0, 0, 0)	If image alignment is enabled, the captured scan will be cropped to a smaller size that fits a rectangle. The unit for crop: [pixel]
scanwidth		Width of scan [pixel]
scanheight		Height of scan [pixels]
images[]	• image01.png • image02.png • image03.png • image04.png • image05.png • image06.png	Array with list of filenames for 2D images captured for this scan. Most sensors have 6 images.
activeheightmap	scan05.tmd	Filename for height map TBD The heigtmap is an internally used 3D file. It is structured the same way

		as the 2D image files, but the pixel location does not represent a brightness level like in the 2D files, instead the pixel value represent the Z-height [mm] at the given XY location
activenormalmap	scan05_nrm.png	Filename for normal map The normal map is an internally used file. It is structured the same way as the 2D image files, but the pixel location does not represent a brightness level like in the 2D files, instead the pixel value represent the surface normal at the given XY location
calib	C:\Users\Public\Documents\GelSight\Scans\scan05\Calib-2A3F-2JTC_20230606_1413.yaml	Path and filename for calibration data

aligned	false	Alignment status (bool)
cameratostage		TBD
calibration.date	2023-06-06 14:14:12	Date of calibration
calibration.username	TABLET-BTIUPLFS\GelSight	Windows username of the account used at time of calibration
camera.cameraid	CIMAF2239054	Camera (device) serial number
camera.cameratype	Ximea	Camera vendor
camera.gelid	2A3F-2JTC	Gel serial number
camera.lensfocuspos	-201.95	Focus setting NOTE: only present on sensors that support autofocus
camera.shutter	0.689	Shutterspeed [ms]
device.deviceconfigid	3	Unique id of the device model
device.devicefirmware	412	Firmware of the device NOTE: not present on all sensor models
device.devicemodel	Series 2	Text

		description of the sensor class
device.devicetemp	50.4	Camera temperature [celcius] NOTE: not present on all sensor models
device.devicetype	0.5X 5 MP	Text description of the sensor model (one sensor model can have multiple device types)
device.serialnumber	CIMAF2239054	Camera (device) serial number
metadata.appname	GelSight.Mobile	Sensor software interface text description
metadata.appversion	3.4.115.0	GSM version used to capture the scan
metadata.gelid	2A3F-2JTC	Gel serial number
metadata.gelusecount	39	The number of scans taken with this gel
metadata.replica	false	Is replica enabled? [bool]

metadata.sdkversion	3.4.115.0	GSM version used to capture the scan
metadata.username	TABLET-BTIUPLFS\GelSight	Windows username of the account used at time of calibration
metadata.detrended	false	Is form removal (detrending) enabled? [bool]
metadata.detrendorder	2	The order of detrending (only valid if 'detrended' is 'true')

AnalysisSaved Event

The Results parameter in the AnalysisSaved event contains JSON formatted data from the scancontext.yaml file located in the subfolder "analysis" inside the captured scan's folder on disk. It is recommended to use the free C# Newtonsoft Json.NET library to help parse this information, but the API developer is free to use any method that they choose.

The JSON string contains two arrays.
- shapes
- routines

The "shapes" contain an entry for each user-created shape drawn on the scan.

The "routines" contain an entry for each analysis method that the user has saved for this scan.

Shape
Shapes are defined in a coordinate system that has (0,0) located in the TopLeft corner of a scan.

Each shape has the following shared properties.

Header	Example	Comment
id	102750541	Unique id referring to this instance of

		shape [int]

Rectangle:

Header	Example	Comment
type	Rectangle	Internal string reference to the type of shape [string]
name	Rectangle	User facing name of the shape [string]
x	1216.79012346	X position of TopLeft corner [pixel]
y	313.32345679	Yposition of TopLeft corner [pixel]
w	562.765432099	Width of rectangle [pixel]
h	1219.83209877	Height of rectangle [pixel]
rotation	0	Rotation [degree]

Ruler:

Header	Example	Comment
type	Ruler	Internal string reference to the type of shape [string]
name	Ruler	User facing name of the shape [string]
x1	1216.79012346	X position of first point of ruler [pixel]
y1	313.32345679	Y position of first point ofruler [pixel]
x2	562.765432099	X position of second point ofruler [pixel]
y2	1219.83209877	Y position of second point ofruler [pixel]

Line:

Header	Example	Comment
type	Line	Internal string reference to the type of shape [string]
name	Line	User facing name of the shape [string]
x1	1216.79012346	X position of first point of line [pixel]
y1	313.32345679	Y position of first point of line [pixel]
x2	562.765432099	X position of second point of line [pixel]

y2	1219.83209877	Y position of second point of line [pixel]

Circle:

Header	Example	Comment
type	Circle	Internal string reference to the type of shape [string]
name	Circle	User facing name of the shape [string]
x	1216.79012346	X position of center [pixel]
y	313.32345679	Y position of center [pixel]
r	562.765432099	radius [pixel]

Polygon/Polyline:

Header	Example	Comment
type	PolyLine	Internal string reference to the type of shape [string]
name	PolyLine	User facing name of the shape [string]
points[]	[(66.2933631391, 1300.69689919), (2348.81804603, 1367.46178936), (62.1205575031, 2001.72824603)]	Array of X/Y position of each point in the PolyLine [pixel]
closed	true	true: polygon false: polyline

Routines

Some routines will allow the user to draw one or more shapes and use them as input to the routine. If so, those inputs are referred to as shapeid which is referencing the array of "shapes". One example is offset which require a user drawn line as shape input

ABS 1781:

Header	Example	Comment
type	FastenerABS1781	Internal string reference to the type of routine [string]
id	1458553698	Unique id referring to this instance of routine

name	ABS 1781	User facing name of the routine [string]
annotationids[]		TBD
diameter	8.75	Input: diameter [mm]
flushoffset	0.5	Input: Flushness offset [mm]
flushminlim	-0.08	Input: Flushness Min Limit [mm]
flushmaxlim	0.1	Input: Flushness Max Limit [mm]
headdishmin	0.158	Output: Head Dishing Min [mm]
perpendicularity	1.97412801048	Output: Perpendicularity [degree]
flushcircle	(1811.29474823, 753.969925059, 524.440141531)	Output: location of circle (X,Y,Radius) [pixel]
flushminpt	(1291, 826)	Output: Location of min point (X,Y) [pixel]. Rendered in UI as blue dot on scan overlay
flushmaxpt	(1291, 826)	Output: Location of max point (X,Y) [pixel]. Rendered in UI as red dot on scan overlay
image	image.png	Output: Filename for scan overlay for UI rendering
passfail	false	Output: was routine able to run, i.e. was enough inputs provided to allow a run [bool]
meta_passedanalysis	false	Output: result of pass/fail settings made by user [bool]
meta_failurereason	Pass value is `False`. Required value is `True`	Output: text description of pass/fail result [string]
meta_sdkversion	3.7.102.0	GSM version used to run this routine

ABS 2322:

Header	Example	Comment
type	FastenerABS2322	Internal string reference to the type of routine [string]
id	176253893	Unique id referring to this instance of routine
name	ABS 2322	User facing name of the routine

		[string]
annotationids[]		TBD
diameter	6.76	Input: diameter [mm]
corediameter	4.7	Input: core diameter [mm]
coreminlim	-0.1	Input: Break-Off Min Limit [mm]
coremaxlim	0.1	Input: Break-Off Max Limit [mm]
flushoffset	0.5	Input: Flushness offset [mm]
flushminlim	-0.08	Input: Flushness Min Limit [mm]
flushmaxlim	0.1	Input: Flushness Max Limit [mm]
flushmin	-0.0752567052841	TBD
flushmax	0.0198004096746	TBD
coremin	-0.654451012611	TBD
coremax	0.0789419040084	TBD
perpendicularity	0.852393517214	Output: Perpendicularity [degree]
flushminpt	(1663, 1067)	Output: Location of flush min point (X,Y) [pixel]. Rendered in UI as blue dot on scan overlay
flushmaxpt	(944, 511)	Output: Location of flush max point (X,Y) [pixel]. Rendered in UI as red dot on scan overlay
coreminpt	(1403, 682)	Output: Location of core min point (X,Y) [pixel]. Rendered in UI as blue dot on scan overlay
coremaxpt	(1131, 534)	Output: Location of core max point (X,Y) [pixel]. Rendered in UI as red dot on scan overlay
image	image.png	Output: Filename for scan overlay for UI rendering
meta_passedanalysis	true	Output: result of pass/fail settings made by user [bool]
meta_failurereason		Output: text description of pass/fail result [string]
meta_sdkversion	3.7.102.0	GSM version used to run this routine

Defect Detection:

Header	Example	Comment
type	DefectDetection	Internal string reference to the type of routine [string]
id	1027624229	Unique id referring to this instance of routine
name	Defect Detection	User facing name of the routine [string]
annotationids[]		TBD
scratchtype36	2	Input: Scratch type [enum] 1=narrow 2=wide 3=large
autothreshold	true	Input: automatic set threshold [bool] (hidden in UI)
depththreshold	0.005	Input: depth threshold [mm] (hidden in UI)
levelwidth	0.35	Input: levelling width [mm] (hidden in UI)
leveldiscardwidth	0.2	Input: Level Discard Width [mm]
levelregions	'[(0, 0.35087546203, None), (1.26590362772, 1.60989917873, None)]'	Input: Array of leveling regions. Each entry has the following properties • Start (referring to position on profile line) [mm] • End (referring to position on profile line) [mm] • Type (None/Max/Min), will always be None for level regions
primaryshapeid	0	Input: Reference to the shapeID used as input. 0=no shape input
debug	false	Internal use
length	3.37853770121 mm	Output: Length of defect [string]
width	1.27010051581 mm	Output: width of defect [string]
area	1.10270439261 mm3	Output: area of defect [string]
depth	0.23071826702 mm	Output: depth of defect [string]

point	(1687.88679927, 768.239506643)	Output: Location of deepest point (x,y) [pixel]
image	image.png	Output: Filename for scan overlay for UI rendering
profile	'[(0, -0.0037239978619), ... 0.0182266798741)]'	Output: Array of points in the profile. Each point in the profile is organized as (X,Y) [mm]
minpt	(0.67423127998, -0.23071826702)	Output: The location and value of the lowest point in the profile (X,Y) [mm]
plot	'[(0.0206397330606, ... -0.109643088634)]'	TBD
profilelines	'[(1300.97348203, ... 873.307388357)]'	Output: Array of 5 lines to be rendered on the scan overlay. They define the area of the defect and the profile line. Each line is defined by (x, y, width, length) [pixel]
meta_passedanalysis	true	Output: result of pass/fail settings made by user [bool]
meta_failurereason		Output: text description of pass/fail result [string]
meta_sdkversion	3.7.102.0	GSM version used to run this routine