i2 V1.1.4 API

USER GUIDE

Version 1.2   March 2019

TABLE OF CONTENTS

INTRODUCTION

This document provides an overview of the MoTeC i2 API Licence installation, activation and general use.

THE i2 API LICENCE

The i2 API Licence allows a user to programmatically interact with logged data via i2 Pro. Some of the functionality of the i2 API Licence includes:

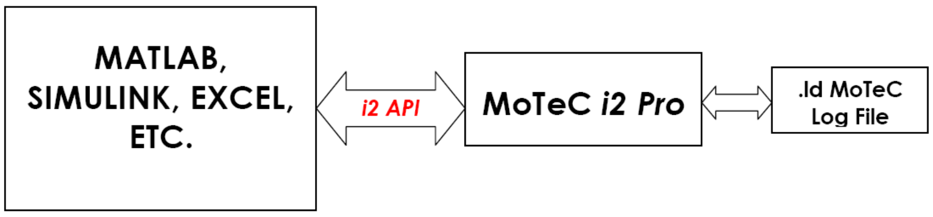
- Loading any MoTeC pro enabled log file (.ld file)
- Extracting data including channels, laps, beacons and details
- Creating new channels programmatically
- Exporting data to i2 Pro log files

ABOUT THE MOTEC i2 API

Users can write their own custom software with any programming language that supports COM technology, such as MS Excel, MATLAB and .NET (VB.NET, C#).

The MoTeC i2 API is an intermediate layer between the i2 application and the third party software that can be used to:

- Get data from an external source, process it and send it to i2; e.g. from simulation software or conversion from another logging format.
- Get data from a MoTeC log file (.ld), then transform that data for another system (e.g. so it can be used as an input to a seven-post rig to replicate the suspension movements of an actual race using real vehicle data).



<u>MoTeC i2 API Samples</u>

When i2 Pro is installed, a folder containing i2 API samples is supplied. The samples are simple, but they outline how to interact with the i2 API in order to get or set data from i2 Pro.

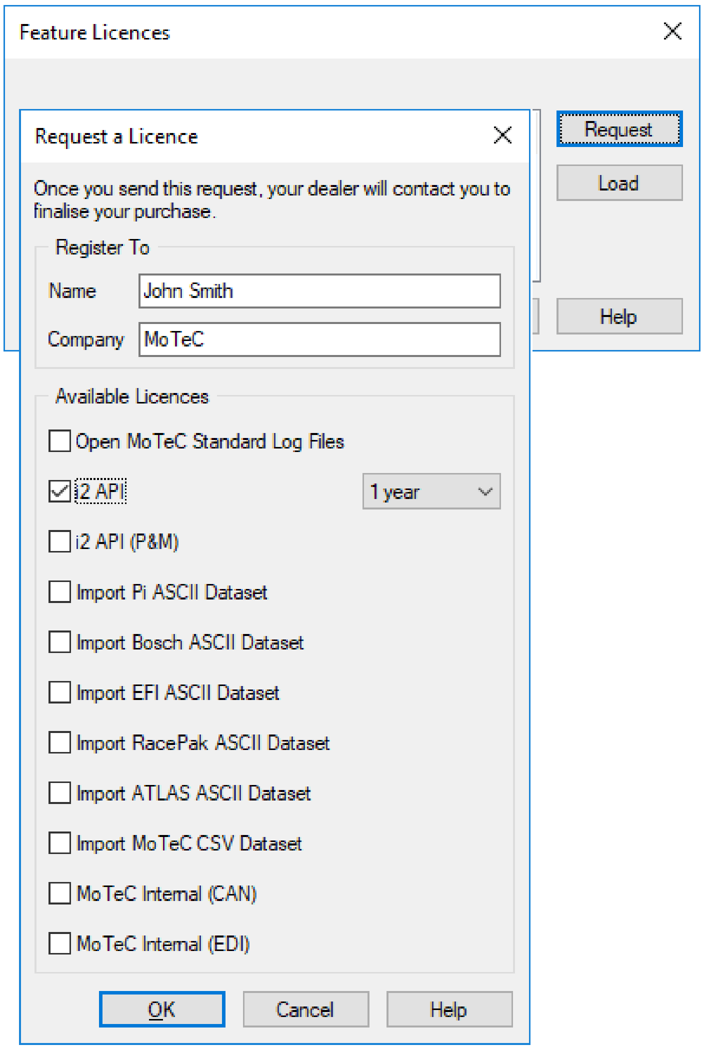
On installation (x64), the sample files are located in the following folder:

C:\Program Files\MoTeC\i2\1.1\Samples\i2API

i2 API INSTALLATION AND ACTIVATION

Installation

1. Install the latest release version of i2 Pro from http://www.motec.com/software/latestreleases/.
2. Start i2 Pro and open a Workspace. If no Workspaces exist in i2 Pro, create a new one via the **File > Workspace > New Workspace…** menu.
3. Once a Workspace is open, go to the **Help > Licences** menu and click on the **Request** button.
4. In the Request a Licence window, enter your Name and Company, select i2 API and select **OK**. See Figure 1.



Once you send this request, your dealer will contact you to finalise your purchase.

Register To

Name	John Smith
Company	MoTeC

Available Licences

- [ ] Open MoTeC Standard Log Files
- [✓] i2 API — 1 year
- [ ] i2 API (P&M)
- [ ] Import Pi ASCII Dataset
- [ ] Import Bosch ASCII Dataset
- [ ] Import EFI ASCII Dataset
- [ ] Import RacePak ASCII Dataset
- [ ] Import ATLAS ASCII Dataset
- [ ] Import MoTeC CSV Dataset
- [ ] MoTeC Internal (CAN)
- [ ] MoTeC Internal (EDI)

Figure 1: Request a Licence example

Note: The i2 Licence is issued for the specific computer and will need to be updated if the computer hardware changes.

If you are using an email client (like Outlook or similar), the Licence request email is created automatically after clicking **Yes** in the Confirmation window.

Note: If there is no email client running on your computer, a file with instructions will open. Create an email by following the instructions given.

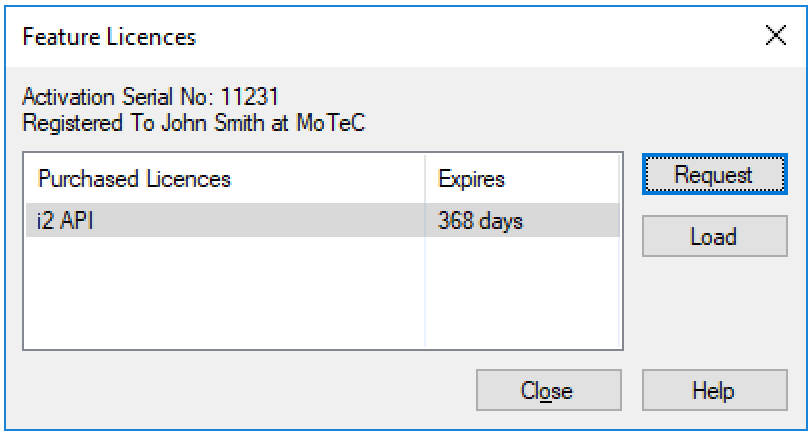
5. Verify that the Licence request file **i2.mtcreq** is attached to the email.
6. Send the email to your MoTeC dealer.

The dealer will respond by sending you a Licence Activation file.

Activation

7. On receiving the Licence Activation file from your dealer, save the file to your computer.
8. Start i2 Pro and open a Workspace.
9. Go to the **Help > Licences** menu. The Feature Licences window displays.
10. Select the **Load** button and select the Licence Activation file.

After the Licence is activated, i2 API appears in the Feature Licences list with the number of days for which the Licence is valid. Ensure the Activation Serial No is shown at the top of the window.


Feature Licences

Activation Serial No: 11231
Registered To John Smith at MoTeC

Purchased Licences	Expires
i2 API	368 days

Figure 2: Feature Licences Window example

USING THE SAMPLES

C#

1. To select the C# sample file, go to: **C:\Program Files\MoTeC\i2\1.1\Samples\i2API\C#**.
2. Double click on the **i2API.sln** file.

   If Visual Studio is installed and associated with ***.sln** files, the sample file will open in the Visual Studio IDE.

   Note: If i2 Pro is installed properly and the i2 API Licence is activated, you will be able to compile and run the sample program.

MATLAB

1. To select the MATLAB sample file, go to: **C:\Program Files\MoTeC\i2\1.1\Samples\i2API\MATLAB**.
2. Double click on the **i2API.m** file.

   If MATLAB is installed and associated with ***.m** files, the sample file will open in the MATLAB editor.

   Note: If i2 Pro is installed properly and the i2 API Licence is activated, you will be able to compile and run the sample program.

Troubleshooting

The most common error relates to MATLAB being unable to create the OLE Automation server for the i2 Pro API.

The error usually occurs at the line **i2 = actxserver('MoTeC.i2Application');** and the error message is something like "ActiveX - Invalid ProgID 'MoTec.i2Application'".

This issue may be caused by:

- incomplete i2 API registration (may require administrator privileges)
- 32/64 bit interop failure

If the error persists, or support is required, please send an email to support@motec.com.au. It is always helpful to attach screenshots and an example of the code that highlights the problem.

Note: By default, the i2 API loads the most recently used i2 Pro Workspace.

i2 NAMESPACE OBJECT DEFINITIONS

Introduction

This section outlines the object and interface definitions within the “i2” namespace.

Note: Parts of the i2 API are also used by i2 Pro External Maths Plug-ins and External Maths Functions which are discussed at the end of this document.

Scope

The definitions below are written against “***MoTeC i2 2.3 Type Library***” unless specified otherwise.

Items marked as “Internal Use Only” are not expected to be used by third parties and are thus not fully documented.

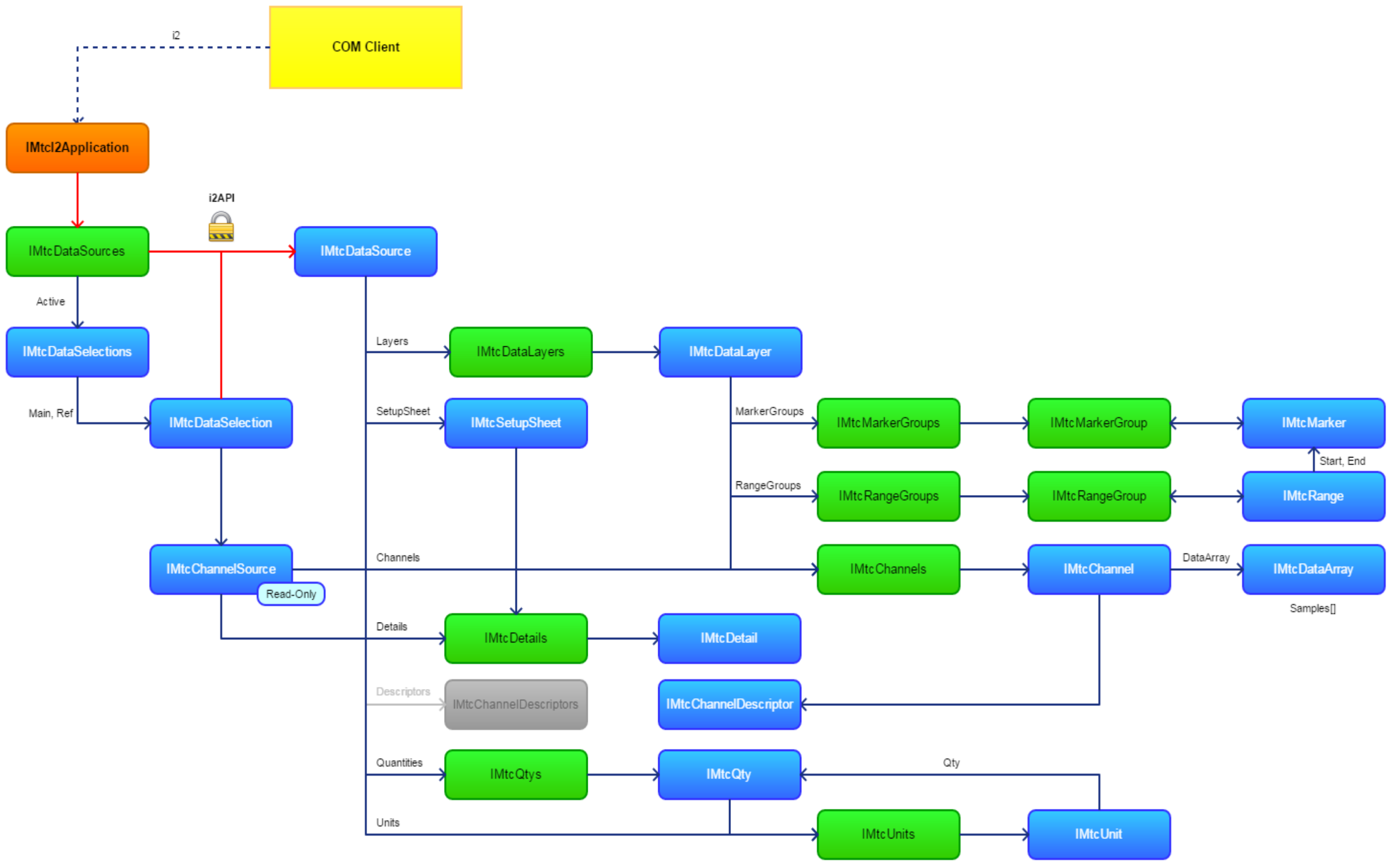
Some programming knowledge is expected. Terms such as Instantiation, Namespace, Class, Method and Property are used throughout this section.

Notes

- Be sure to add the latest i2 Type Library as a reference to all .NET based projects.
- All time values (unless stated otherwise) are in microseconds [µs].

OVERVIEW

The following diagram illustrates the main interfaces and connections associated with the i2 API.



TYPES

The following enumerations are used by the classes and interfaces described throughout this document.

AntiAliasFilterType

- aaNone
- aaAvg
- aaMin
- aaMax

This type is only available with i2 Type Library v2.4 or higher.

ChannelStatus

- Valid
- Invalid

Dimension

- Array
- Scalar

EDataExtent

- deMainRange
- deMainRangeOuting
- deMainRangeZoom

ESampleRateOpt

- srCustom
- srDefault
- srFastest

The following types provide string constants to be used by some API methods:

DataType

Integer	= *"MoTeC.Integer"*
Real	= *"MoTeC.Float"*
String	= *"MoTeC.String"*

Layer

Burst0	= *"Burst0"*
Burst1	= *"Burst1"*
Details	= *"Details"*
Fastest	= *"Fastest"*
Normal	= *""*
SetupSheet	= *"SetupSheet"*

CLASSES

The following classes represent the primary entry point into the i2 API. Once instantiated, these instances will provide the ability to create, read and update data within i2.

Application

Class denoting an instance of the i2 application. It connects to an already running instance of the i2 application, or starts one if not.

Example (C#)

var i2 = new i2.Application();
i2.Visible = true;
i2.Exit();

Implements the ***IMtcI2Application*** interface.

ApplicationSingle

Class denoting an instance of the i2 application. Unlike the *Application* class, it always creates a new instance of an i2 application. It implements same properties and methods as the *Application* class.

Example (C#)

var i2 = new i2.ApplicationSingle();
i2.Visible = true;
i2.Exit();

Implements the ***IMtcI2Application*** interface.

INTERFACES

The following interfaces, in combination, make up what is known as the i2 API.

IMtcApplication

The generic MoTeC application interface.

Properties

Name	Type	Description
Name	string	Get the application name
Version	string	Get the application version
Visible	boolean	Get the application's visibility. COM Server usually starts invisible and must be explicitly made visible

Methods

Name	Parameters Description
void **Exit()** Exit the i2 application	
void **ShowHelp()** Show the i2 Help Window	
void **ShowHelpAbout()** Show the i2 Help About window	

IMtcChannel

Represents a channel in i2.

Properties

Name	Type	Description
DataArray	IMtcDataArray	Get/Set the data array (samples) associated with this channel
Descriptor	IMtcChannelDescriptor	Get the descriptor for this channel
Id	string	Get the channel display name
Status	*ChannelStatus*	Get/Set the data array validity status
StatusText	string	Get/Set the data array validity message. This text is displayed in the "**Tools|Channel Status**" dialogue of i2

Methods

Name	Parameters Description
IMtcDataArray **CreateDataArray**(double *SampleRate*) Creates a DataArray for this channel definition, based on the sample rate required	*SampleRate*: usually between 1 – 1000 Hz A buffer is created with enough space to store samples at this rate for the time extent of the data source
void **SetData**(IMtcDataArray *DataArray*, ChannelStatus *Status*, string *StatusText*) Updates the channel's data state	*DataArray*: the new data to be set *Status*: the new channel status to be set *StatusText*: the new message to be set. When the status is invalid, it can indicate the reason

IMtcChannel2

Extends the IMtcChannel interface with the following properties. These properties are a shortcut to the same properties exposed via the **Descriptor**.

Properties

Name	Type	Description
Color	long	Get the color (RGB)
ColorIndex	long	Get/Set the color index. This specifies a color as an index into the color theme palette for channels (see "**Tools|Options|Colors**" in i2)
DPS	long	Get/Set the number of decimal places to be shown for any value
Interpolate	boolean	Get/Set the interpolation mode for the channel data. When true, the linear interpolation is used, otherwise previous value is retained
ScaleMin	double	Get/Set the minimum expected value. Its units must be based on the current "Unit" settings
ScaleMax	double	Get/Set the maximum expected value. Its units must be based on the current "Unit" settings
Unit	string	Get/Set the display unit. The unit string must match an i2 text unit as shown in "**Tools|View Units…**"

IMtcChannel3

Extends the IMtcChannel2 interface with additional properties and methods.

This type is only available with i2 Type Library v2.4 or higher.

Methods

Name	Parameters Description
[IMtcDataArray](#) ***DataArraySegmentFromIndexes*** (int *StartIndex*, int *Count*, double *SampleRate*, int *AntiAliasFilterType*) Generates a resampled data array over the interval defined by the start/count indexes	*StartIndex*: an index to start resampling *Count*: the number of samples to be resampled *SampleRate* [Hz]: the new sampling rate (can be higher or lower than the original rate) *AntiAliasFilterType* [enum]: when decimating, it is applied to the source samples
[IMtcDataArray](#) ***DataArraySegmentFromInterval*** (double *StartTime*, double *EndTime*, double *SampleRate*, int *AntiAliasFilterType*) Generates a resampled data array over the interval defined by the start/end times	*StartTime* [µs]: time to start resampling *EndTime* [µs]: time to stop resampling *SampleRate* [Hz]: the new sampling rate (can be higher or lower than the original rate) *AntiAliasFilterType* [enum]: when decimating, it is applied to the source samples

IMtcChannelDescriptor

Defines some basic properties that describe a channel (without the channel necessarily existing).

Properties

Name	Type	Description
Color	long	Get the color (RGB)
ColorIndex	long	Get/Set the color index. This specifies a color as an index into the color theme palette for channels (see "**Tools|Options|Colors**" in i2)
DPS	long	Get/Set the number of decimal places to be shown for any value

Name	Type	Description
Id	string	Get the display name
Interpolate	boolean	Get/Set the interpolation mode for the channel data. When true, the linear interpolation is used, otherwise previous value is retained
ScaleMin	double	Get/Set the minimum expected value. Its units must be based on the current "Unit" settings
ScaleMax	double	Get/Set the maximum expected value. Its units must be based on the current "Unit" settings
Unit	string	Get/Set the display unit. The unit string must match an i2 text unit as shown in "**Tools|View Units…**"

IMtcChannelDescriptors

A collection of **IMtcChannelDescriptor** objects.

Properties

Name	Type	Description
Count	long	Get the number of arguments in the collection

Methods

Name	Parameters Description
IMtcChannelDescriptor ***Add*** (string *Id*, string *Type*, string *Unit* **)** Add a new descriptor to the collection. The Id must be unique	*Id*: display name *Type*: the ***DataType*** of the channel *Unit*: the default display unit
void ***Remove*** (string *Id*) Remove the named channel descriptor from the collection	*Id*: display name
IMtcChannelDescriptor **Index[**long *Nth***]** Get the Nth descriptor	*Nth*: arguments index
IMtcChannelDescriptor **Item(**string *Name***)** Get the named descriptor	*Name*: arguments name (same as Id)

IMtcChannels

A collection of **IMtcChannel** objects.

Properties

Name	Type	Description
Count	long	Get the number of channels in the collection

Methods

Name	Parameters Description
IMtcChannel ***Add*** (string *Id*, string *Type*) Add a new channel to the collection. The Id must be unique	*Id*: display name *Type*: the ***DataType*** of the channel
IMtcChannel **Index[**long *Nth***]** Get the Nth channel	*Nth*: arguments index
IMtcChannel **Item(**string *Name***)** Get the named channel	*Name*: arguments name

IMtcChannelSource

Represents the channels available from a data source. All interfaces from here are read-only.

Properties

Name	Type	Description
Channels	IMtcChannels	Get the channels associated with this source
Details	IMtcDetails	Get the details associated with this source
Name	string	Get the name of this source

IMtcDataArray

The main interface for dealing with data samples in i2.

Properties

Name	Type	Description
Count	long	Get the number of samples of the data
EndTime	double	Get the end time (μs) of the data
Rate	double	Get the sample rate (Hz) of the data
SampleFlags	object [array]	Get/Set the actual sample flags. Each sample contains a flag indicating validity: • 0 = invalid • 1 = valid
Samples	object [array]	Get/Set the actual samples
StartTime	double	Get the start time (μs) of the data
Unit	string	Get/Set the data unit. The unit string must match an i2 text unit as shown in "**Tools|View Units…**" **Note:** This does not perform any unit conversion but simply labels the data as being in this unit

Methods

Name	Parameters Description
Object ***GetValue*** (double *Time*) Return the sample value at the given time. Depending on the channel's interpolation mode, it may return an interpolated value	*Time* [µs]: time
long ***SampleIndex*** (double *Time*) Return the sample index for a given time	*Time* [µs]: time
double ***SampleTime*** (long *Index*) Return the sample time [µs] for a given index. The index may be beyond the interval of the actual samples available	

IMtcDataLayer

Represents a time interval of logging. Typically used to represent Normal and Burst logging groups.

Properties

Name	Type	Description
Channels	IMtcChannels	Get the channels collection
MarkerGroups	IMtcMarkerGroups	Get the marker groups collection
Name	string	The name of the layer (see Layer) An empty name implies "Normal"
RangeGroups	IMtcRangeGroups	Get the range groups collection

IMtcDataLayers

A collection of **IMtcDataLayer** objects.

Properties

Name	Type	Description
Count	long	Get the number of data layers in the collection

Methods

Name	Parameters Description
IMtcDataLayer **Index[**long *Nth***]** Get the Nth data layer	*Nth*: data layer index
IMtcDataLayer **Item(**string *Name***)** Get the named data layer	*Name*: data layer name. Valid names are defined in the Layer enumeration

IMtcDataSelection

Represents a data selection in i2.

Properties

Name	Type	Description
CursorTimeDatum	double	Get the datum cursor time [µs]
CursorTimeMain	double	Get the main cursor time [µs]
CursorTimeRealTime	double	Get the real-time cursor time [µs]. Only available when the data selection is tracking a live telemetry data source
DataSource	IMtcDataSource	Get the active data source
Name	string	Get the data selection name

Methods

Name	Parameters Description
void **GetZoomTime(** double *StartTime*, [out] double *EndTime* [out] **)** Get the current zoom start/end extent	*StartTime*: out parameter, receives the zoom start time [µs] *EndTime*: out parameter, receives the zoom end time [µs]

IMtcDataSelection2

Extends the IMtcDataSelection interface with the following properties.

Properties

Name	Type	Description
ChannelSource	IMtcChannelSource	Get the associated channel source

IMtcDataSelections

Represents the main and (optional) reference IMtcDataSelection interfaces currently active in i2.

Properties

Name	Type	Description
Main	IMtcDataSelection	Get the main data selection for the active component in i2
Reference	IMtcDataSelection	Get the reference data selection for the active component in i2. This may be null when no reference is selected.
Name	string	Get the name

IMtcDataSource

A data source can represent:

- An LD representing logged data
  or
- An active telemetry stream

Properties

Name	Type	Description
Date	date	Get the date of the data source
Descriptors	IMtcChannelDescriptors	No longer used by i2
Details	IMtcDetails	Get the details associated with the data source
Layers	IMtcDataLayers	Get the data layers existing within the data source
Name	string	Get the name of the data source
Quantities	IMtcQtys	Get the quantities available in i2
SetupSheet	IMtcSetupSheet	Get the setup sheet associated with the data
Time	date	Get the time of this data source
Units	IMtcUnits	Get the units available in i2

Methods

Name	Parameters Description
IMtcDataArray **CreateDataArray(** string *Type*, double *StartTime*, double *EndTime*, double *SampleRate* **)** Create an unbound data array	*Type*: the **DataType** of the array. One of: DataType.Integer DataType.Real DataType.String
	StartTime [µs]: start time of data (usually 0)
	EndTime [µs]: end time of data
	SampleRate [Hz]: sample rate of the data

IMtcDataSources

A collection of **IMtcDataSource** objects.

This interface and methods can only be accessed if you have a valid i2 API Licence

Properties

Name	Type	Description
Count	long	Get the number of data sources in the collection
Main	IMtcRange	Get the currently active main range
Reference	IMtcRange	Get the currently active reference range

Methods

Name	Parameters Description
IMtcDataSource **Index[**long *Nth***]** Get the Nth data layer	*Nth*: data layer index
IMtcDataSource **Open(** string *Source* **)** Open a new data source into i2	*Source*: A fully qualified file name to an LD file, or the telemetry source, prefixed with **t2://**
void **Close(** IMtcDataSource *DataSource* **)** Close the supplied data source	
void **CloseAll()** Close all data sources currently loaded	

Name	Parameters Description
IMtcDataSource **Create(** double *Duration* **)** Create a data source that represents an interval of time	*Duration* [μs]: the amount of time represented by the data source
void **ExportMain(** string *FileName*, EDataExtent *DataExtent* **)** Export the main selection logged channels as an LD file **Note:** Maths channels are not included	*FileName*: the destination file name of the export *DataExtent*: amount of data to be exported (for example, the selected lap, the entire outing or just the currently zoomed extent)
void **ExportMainAsMAT(** string *FileName*, EDataExtent *DataExtent* ESampleRateOpt *SampleRateOption* double *CustomSampleRate* **)** Export the main selection logged channels as a MATLAB (type 5) file **Note:** Maths channels are not included	*FileName*: the destination file name of the export *DataExtent*: amount of data to be exported (for example, the selected lap, the entire outing or just the currently zoomed extent) *SampleRateOption*: specify a sample rate export option *CustomSampleRate*: specify a sample rate if SampleRateOption is set to ***srCustom***
void **Refresh(** IMtcDataSource *DataSource* **)** Refresh any derived details and UI for this data source. Typically it is called after details have been updated.	*DataSource*: the data source to be updated

IMtcDataSources2

Extension of the **IMtcDataSource** interface.

This type is only available with i2 Type Library v2.4 or higher.

Methods

Name	Parameters Description
void **ExportMainAsCSV(** string *FileName*, EDataExtent *DataExtent* **)** Export the main selection logged channels as a MoTeC CSV file **Note:** Maths channels are not included	*FileName*: the destination file name of the export *DataExtent*: amount of data to be exported (for example, the selected lap, the entire outing or just the currently zoomed extent)

IMtcDetail

Represents a detail (named value).

Properties

Name	Type	Description
Id	string	Get the name of the detail
Unit	string	Get the unit of the numeric value
Value	object	Get/Set the value. It will either be a string or a double based on the *IsNumeric* property

Methods

Name	Parameters Description
string **ToString()** Get the value as a string	
bool **IsNumeric()** Returns if the value is interpreted as a number or string	

IMtcDetails

Represents a collection of **IMtcDetail** objects.

Properties

Name	Type	Description
Count	long	Get the number of details in the collection

Methods

Name	Parameters Description
IMtcDetail **Index[**long *Nth***]** Get the Nth detail	*Nth*: details index
IMtcDetail **Item(**string *Name***)** Get the named detail	*Name*: details name
void **AddDateTime(** string *Id,* date *Value* **)** Add a date detail	*Id*: details name *Value*: the date or time value

Name	Parameters Description
void **AddNumeric(** string *Id*, double *Value*, string *Unit*, int *DPS* **)** Add a numeric detail	*Id*: details name *Value*: the numeric value *Unit*: the i2 unit *DPS*: the decimal places
void **AddString(** string *Id*, string *Value* **)** Add a string detail	*Id*: details name *Value*: the string value

IMtcI2Application

The primary entry point into using the i2 API (extends the **IMtcApplication** interface).

Properties

Name	Type	Description
QueryAPI	IMtcQueryAPI	Internal Use Only
DataSources	IMtcDataSources	Get the data sources interface. You must have a valid i2 API Licence in order to retrieve this interface

Methods

Name	Parameters Description
void **WorkspaceNew()** Launch the new Workspace dialogue	
void **WorkspaceOpen()** Launch the open Workspace dialogue	

Name	Parameters Description
void **WorkspaceLoad(** string *File* **)** Load the Workspace from the file	*File*: fully qualified file name to be loaded
void **WorkspaceLoadRecent()** Load the most recently used Workspace	
void **WorkspaceLoadTemplate(template)**	Internal Use Only
void **CheckForUpdates()** Launch the checks for update dialogue	

IMtcMarker

Represents a point in time or distance within the data (layer).

Properties

Name	Type	Description
ClassName	string	Get the marker class. Marker classes differ depending on the type of Workspace that opens. Examples (for circuit) include: • **BCN** (main lap beacon) • **SPLTBCN** (split beacon) • **IGRDBCN** (ignored beacon) • **RESET** (device reset) • **SOL** (start of logging) • **EOL** (end of logging)
Comment	string	Get the comment associated with the marker

Name	Type	Description
MarkerGroup	*IMtcMarkerGroup*	Get the marker group that contains the marker
Name	string	Get the unique name of the marker
Parent	*IMtcMarker*	Get the parent marker. Markers are stored relative to their parent
Time	double	Get the absolute time of the marker [µs]

IMtcMarkerGroup

Represents a collection of **IMtcMarker** (usually of the same class).

Properties

Name	Type	Description
Count	long	Get the number of markers in the collection
Name	string	Get the name of the group

Methods

Name	Parameters Description
IMtcMarker **Index[**long *Nth***]** Get the Nth marker	*Nth*: marker index
IMtcMarker **Item(**string *Name***)** Get the named marker	*Name*: marker name
IMtcMarker **Add(**string *Name*) Add a new marker	*Name*: marker name

Name	Parameters Description
void **Clear()** Clear all the markers from the group	

IMtcMarkerGroup2

Extends the **IMtcMarkerGroup** interface with the following methods.

Methods

Name	Parameters Description
IMtcMarker **AddDistMarker(** IMtcMarker *Parent,* string *ClassName* string *Name,* double *Distance,* string *Comment* **)** Add a distance [m] based marker (relative to the parent marker) to the group	*Parent*: parent marker *ClassName*: the class of marker *Name*: unique name of the marker *Distance* [m]: distance relative to the parent marker *Comment* [optional]
IMtcMarker **AddTimeMarker(** IMtcMarker *Parent,* string *ClassName* string *Name,* double *Distance,* string *Comment* **)** Add a time [µs] based marker (relative to the parent marker) to the group	*Parent*: parent marker *ClassName*: the class of marker *Name*: marker name *Time* [µs]: time relative to the parent marker *Comment* [optional]

IMtcMarkerGroups

A collection of **IMtcMarkerGroup** objects

Properties

Name	Type	Description
Count	long	Get the number of marker groups in the collection

Methods

Name	Parameters Description
IMtcMarkerGroup **Index[**long *Nth***]** Get the Nth marker group	*Nth*: marker group index
IMtcMarkerGroup **Item(**string *Name***)** Get the named marker group	*Name*: marker group name

IMtcRange

Represents a range (defined between two markers) in i2.

Properties

Name	Type	Description
Abbrev	string	Get/Set the range name abbreviation (e.g. “L1” for “Lap 1”)
Enabled	boolean	Get/Set the enable state of the range
Start	IMtcMarker	Get the range start marker
End	IMtcMarker	Get the range end marker
Name	string	Get the range name (e.g. “Lap 1”)
RangeGroup	IMtcRangeGroup	Get the range group this range is associated with
Trusted	boolean	Get/Set the trust state of the range. Untrusted ranges are optionally excluded from i2 calculations (e.g. In/Out laps for auto scale)

IMtcRangeGroup

A collection of **IMtcRange** objects.

Properties

Name	Type	Description
Count	long	Get the number of ranges in the collection
Name	string	Get the name of the range group
SupportsReporting	boolean	Get/Set if report generation can be performed for ranges within the group
SupportsTrackGeneration	boolean	Get/Set if track generation can be performed for ranges within the group

Methods

Name	Parameters Description
IMtcRange **Index[**long *Nth***]** Get the Nth range	*Nth*: range index
IMtcRange **Item(**string *Name***)** Get the named range	*Name*: range name (e.g. “Lap 1”)
IMtcRange **Add(** IMtcMarker *Start*, IMtcMarker *End*, string *Name*) Add a new range	*Start*, *End*: markers defining the extent of the range *Name*: range name
void **Clear()** Clear all the ranges from the group	

IMtcRangeGroups

A collection of **IMtcRangeGroup** objects.

Properties

Name	Type	Description
Count	long	Get the number of range groups in the collection

Methods

Name	Parameters Description
IMtcRangeGroup **Index[**long *Nth***]** Get the Nth range group	*Nth*: range group index
IMtcRangeGroup **Item(**string *Name***)** Get the named range group	*Name*: range group name

IMtcSetupSheet

Represents a Microsoft Excel based setup sheet in i2.

Properties

Name	Type	Description
Details	IMtcDetails	Get the details from the setup sheet
FileName	string	Get the file name of the setup sheet

IMtcQty

Represents a quantity in i2.

Properties

Name	Type	Description
DisplayName	string	Get the quantity name (e.g. Acceleration)
Symbol	string	Get the quantity symbol (e.g. m/s^2)

Name	Type	Description
Units	IMtcUnits	Get the units associated with the quantity

IMtcQtys

A collection of **IMtcQty** objects.

Properties

Name	Type	Description
Count	long	Get the number of quantities in the collection

Methods

Name	Parameters Description
IMtcQty **Index[**long *Nth***]** Get the Nth quantity	*Nth*: quantity index
IMtcQty **Item(**string *Name***)** Get the named quantity	*Name*: quantity name

IMtcUnit

Represents a unit in i2. The full set of units available in i2 can be seen in "**Tools|View Units…**".

Properties

Name	Type	Description
DisplayName	string	Get the unit name (e.g. Celsius)
DisplaySymbol	string	Get the unit symbol (e.g. ^{o}C)
Symbol	string	Get the text-only symbol (e.g. C)
Qty	IMtcQty	Get the quantity this unit belongs to

Methods

Name	Parameters Description
void **FromSI(** object *Value* [in/out] **)** Convert value from SI to this unit	*Value*: [in] SI value, [out] unit value
void **ToSI(** object *Value* [in/out] **)** Convert value from this unit to SI	*Value*: [in] unit value, [out] SI value

IMtcUnits

A collection of **IMtcUnit** objects.

Properties

Name	Type	Description
Count	long	Get the number of units in the collection

Methods

Name	Parameters Description
IMtcUnit **Index[**long *Nth***]** Get the Nth unit	*Nth*: unit index
IMtcUnit **Item(**string *Name***)** Get the named unit	*Name*: unit name

EXAMPLE

The following C# example iterates through channels from a pre-loaded data source in i2 and output name and display unit information.

For the "Engine RPM" channel (if it exists), it will also output 2 seconds of samples from the start of the data.

Example (C#)

```
static double SecondsToTimeBase = 1e6;
static double TimeBaseToSeconds = 1/SecondsToTimeBase;

// Print out all channels in the first data source loaded,
// For "Engine RPM" channel, print out the samples covering the first 2 seconds
var i2 = new i2.Application();
i2.Visible = true;
if (i2.DataSources != null && i2.DataSources.Count > 0)
{
       var ds = i2.DataSources[0];
       var dl = ds.Layers[Layer.Normal];
       if (dl != null)
       {
              foreach (IMtcChannel2 channel in dl.Channels)
              {
                     Console.Write("name = {0,30}, display unit = {1,6}\n", channel.Id,
channel.Unit);

                     if (channel.Id == "Engine RPM") {
                            var da = channel.DataArray;
                            var s = da.Samples as double[];
                            var i0 = da.SampleIndex(0 * SecondsToTimeBase);
                            var i1 = da.SampleIndex(2 * SecondsToTimeBase);
                            for (var i = i0; i < i1; i++)
                            {
                                   var v = s[i];
                                   var t = da.SampleTime(i) * TimeBaseToSeconds;
                                   Console.Write("t = {0,5} v = {1,8} {2,6}\n", t, v, da.Unit);
                            }
                     }
              }
       }
}
i2.Exit();
```

Note: You may need to set the "Embed Interop Types = False" property on the i2 Reference in your Visual Studio solution.

APPENDIX

i2 Math API

Unlike the i2 API interfaces mentioned previously, the i2 Math API are a set of interfaces that you can implement to provide i2 with hooks into your own custom code.

Note: Math plug-ins may need to be registered under administrative privileges (i.e. start i2 with 'Run as Administrator').

IMtcArgs

Represents the i2 Math function arguments passed into your custom Math function.

Properties

Name	Type	Description
Count	long	Get the number of arguments in the referenced collection

Methods

Name	Parameters Description
IMtcMathArray **CreateResultArray()** Creates a suitable Math array to store any generated samples.	
object **Index**[long *Nth*] Get the Nth argument	*Nth*: arguments index
object **Item(**string *Name***)** Get the named argument	*Name*: arguments name

IMtcMathArray

Extends the **IMtcDataArray** interface with additional properties.

Properties

Name	Type	Description
MathEndTime	double	Get the end time for this Math based array [µs]
MathStartTime	double	Get the start time for this Math based array [µs]

IMtcMathFunction

Implement this interface to add your own Math functions into i2.

Methods

Name	Parameters Description
object **Evaluate(** IMtcArgs *Args* **)** Implement this method and return back with a single value or a data array	*Args*: The arguments collection passed into this function

IMtcMathPlugin

Implement this interface to add your own Math results into i2.

Properties

Name	Type	Description
InputChannels	string [array]	Get the channel names this plug-in requires
OutputChannels	string [array]	Get the channel names this plug-in generates
Settings	string	Get/Set the plug-in settings as a string (e.g. JSON)

Name	Type	Description
Summary	string	Get the high level description of the plug-in

Methods

Name	Parameters Description
object **Execute(** IMtcMathPluginArgs *Args*, IMtcChannels *Results* **)** Implement this method and perform whatever calculations you require, returning to i2 multiple results	*Args*: The arguments collection passed into this plug-in *Results*: the collection of channels that the plug-in generates
void **Register(** IMtcChannelDescriptors *Descs* **)** Implement this method to register the details of the channels this plug-in generates	*Descs*: Register the name and type (color etc.) of the channels that are to be generated Since i2 generates Maths on demand, plug-ins must register their intent to generate channels before they are required to actually perform the calculations
void **UnRegister(** IMtcChannelDescriptors *Descs* **)** Implement this method to unregister the channels this plug-in generates	
void **ShowSettings()** Implement this method to show any UI required for the user to edit settings	

IMtcMathPluginArgs

Represents the i2 Math plug-in arguments.

Properties

Name	Type	Description
DataLayerName	string	Get the data layer name this plug-in is currently executing in (within the supplied data source context)
DataSource	IMtDataSource	Get the data source context this plug-in is currently executing in

Range Groups

Range groups are collections of ranges. A range defines a time or distance extent (based on markers) within the normal data layer.

The following range groups are available in i2 (based on their Workspace type):

Circuit

"**Outings:Laps**" – A collection of lap ranges

"**Outings:Laps:Splits**" - A collection of splits

Drag

"**Outings:Runs**" – Typically a single run range

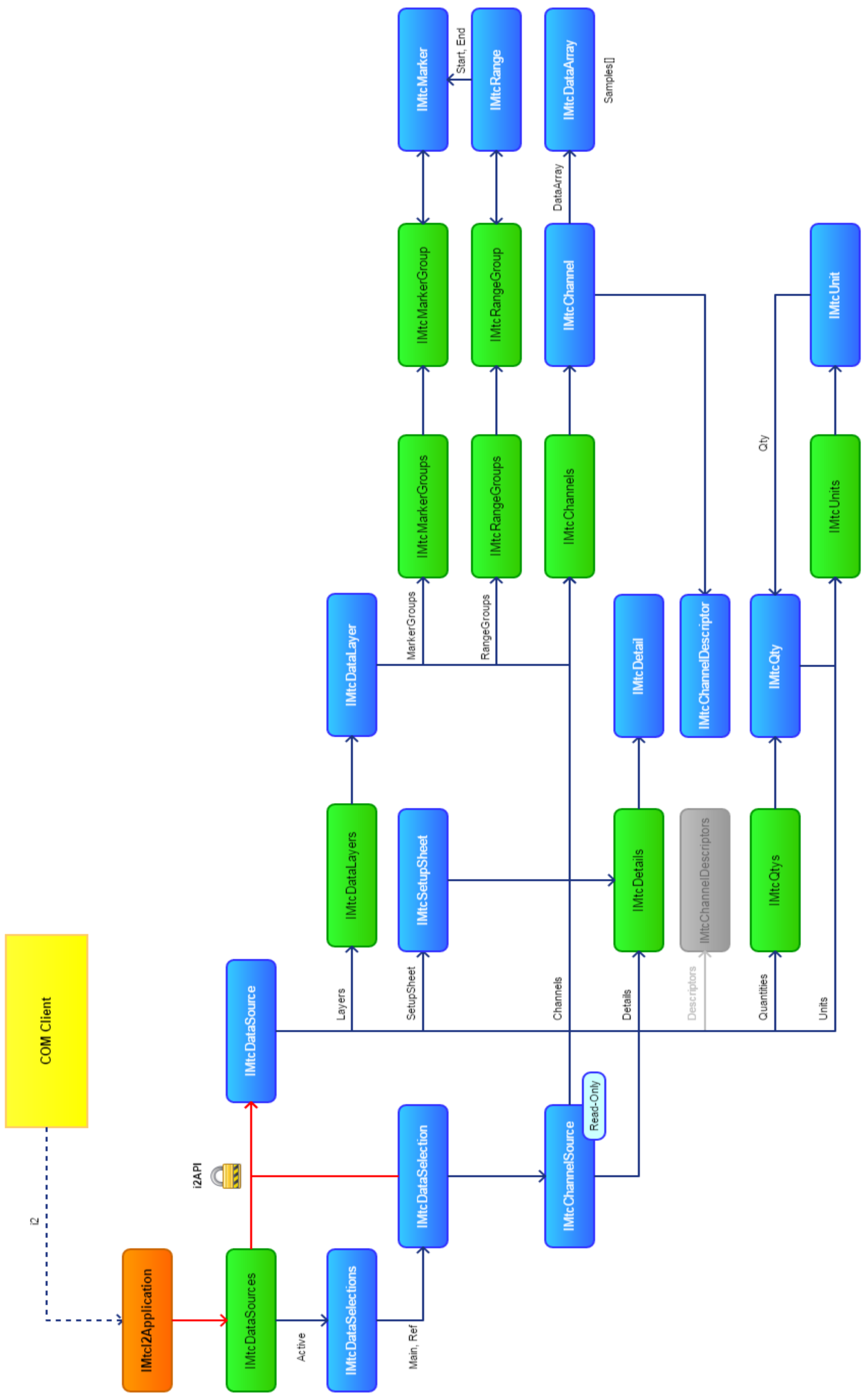
Rally

"**Outings:Stages**" – Stage and transport ranges

Engine

"**Outings:Data**" – Typically a single extent of data

Object Model Reference


COM Client
i2
IMtcI2Application
IMtcDataSources
i2API
IMtcDataSource
Active
IMtcDataSelections
Main, Ref
IMtcDataSelection
IMtcChannelSource
Read-Only
Layers
SetupSheet
Channels
Details
Descriptors
Quantities
Units
IMtcDataLayers
IMtcSetupSheet
IMtcDetails
IMtcChannelDescriptors
IMtcQtys
IMtcDataLayer
IMtcDetail
IMtcChannelDescriptor
IMtcQty
MarkerGroups
RangeGroups
IMtcMarkerGroups
IMtcRangeGroups
IMtcChannels
IMtcUnits
Qty
IMtcMarkerGroup
IMtcRangeGroup
IMtcChannel
IMtcUnit
DataArray
IMtcMarker
IMtcRange
IMtcDataArray
Start, End
Samples[]