SynapseAPI Manual

Updated 2025-02-27

Tucker-Davis Technologies
11930 Research Circle
Alachua, FL 32615 USA
Phone: +1.386.462.9622
Fax: +1.386.462.5365

Notices

The information contained in this document is provided "as is," and is subject to being changed, without notice. TDT shall not be liable for errors or damages in connection with the furnishing, use, or performance of this document or of any information contained herein.

The latest versions of TDT documents are always online at https://www.tdt.com/docs/

Table of Contents

SynapseAPI

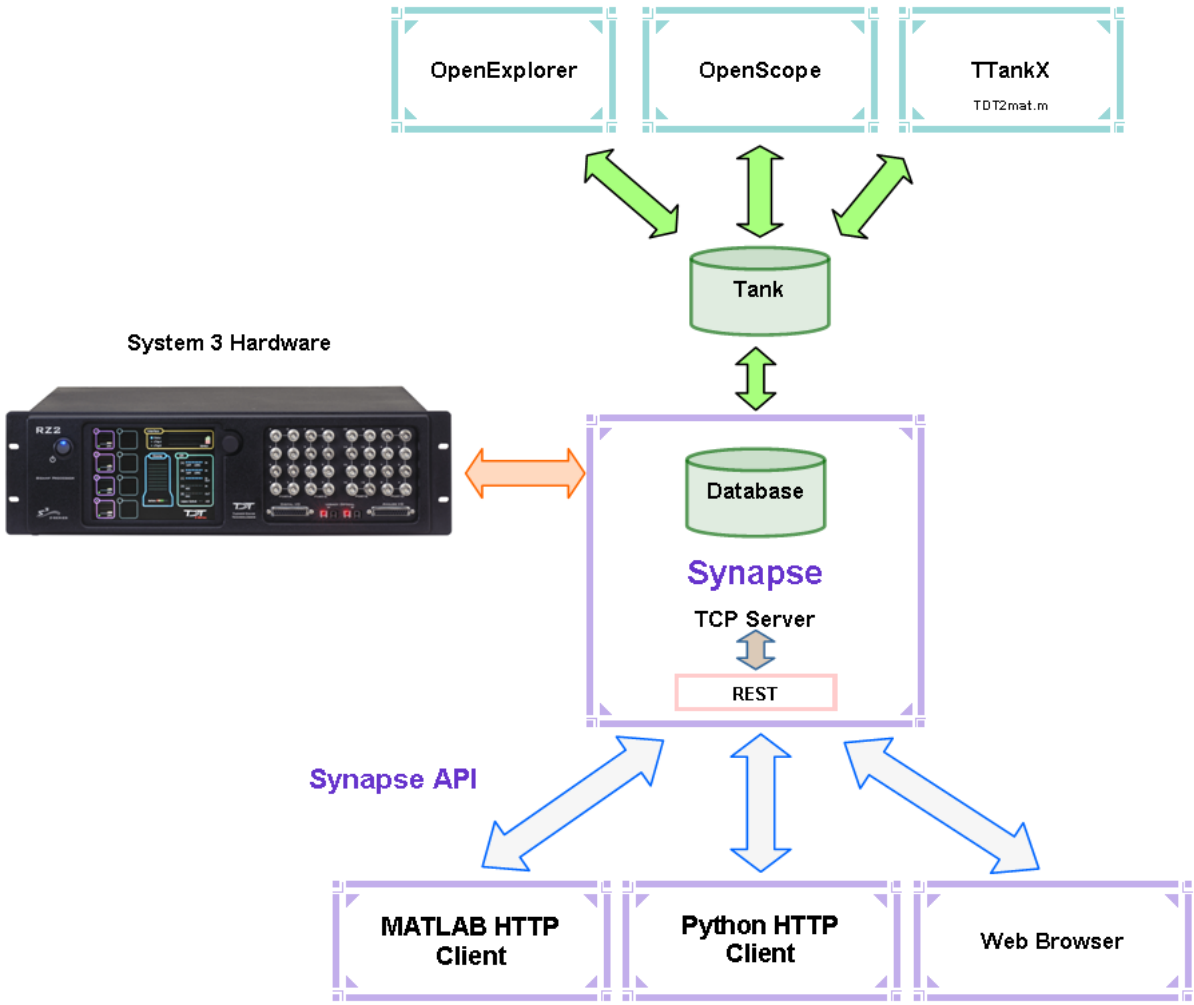
SynapseAPI Overview

SynapseAPI is a collection of methods used to develop Synapse companion applications using MATLAB, Python, C++ or any language that can make HTTP requests. It provides access to System 3 real-time processing devices during an experiment, and lets you control Synapse remotely, mimicking many behaviors a user can perform.

Client applications developed using SynapseAPI can:

- Control the system mode
- Set tank and block names
- Set subject, user, and experiment, and add entries to the Synapse database
- Read and write gizmo parameters at runtime, including user gizmos
- Issue triggers and retrieve system status

SynapseAPI is a RESTful interface between a built-in TCP server and user development tools. Client applications can connect to Synapse through direct HTTP requests.

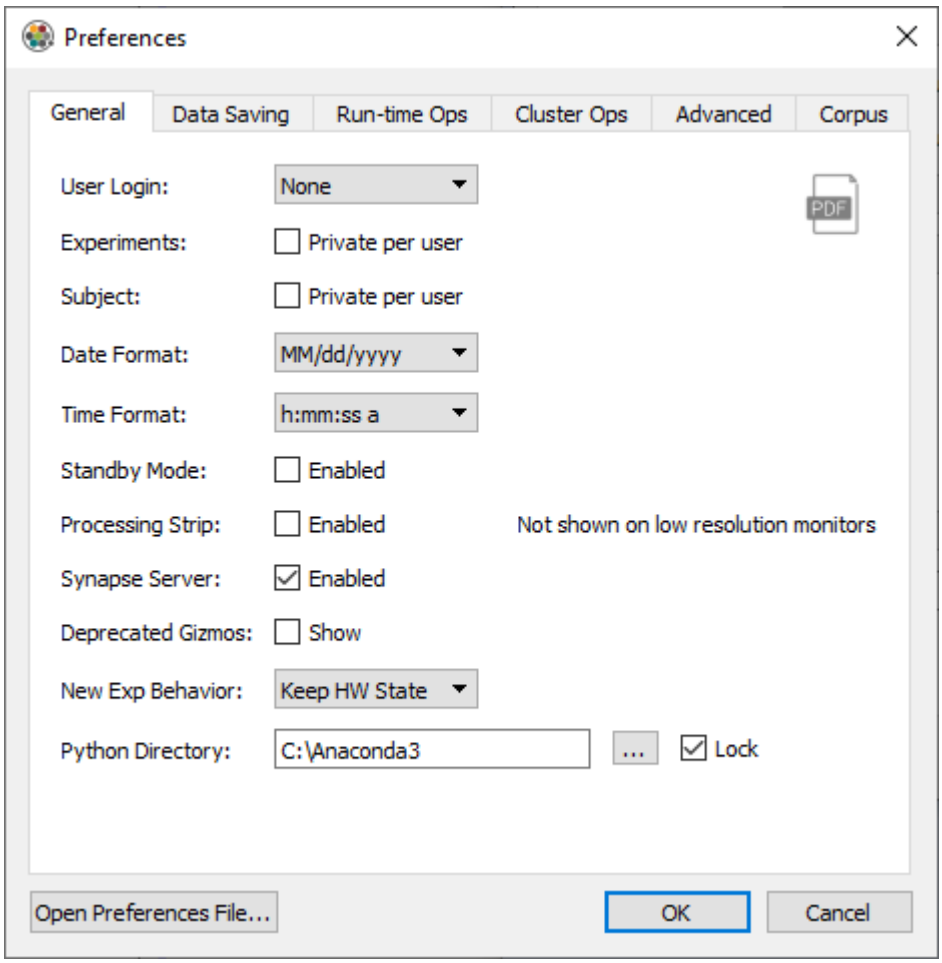


Synapse also uses TDT's TTank data server and data format which provides compatibility with many of the TDT OpenEx client applications, such as OpenExplorer and OpenScope, and includes direct data import into MATLAB with TDTbin2mat or Python's read_block for offline analysis or TDT2mat for online analysis.

Enable SynapseAPI in Synapse

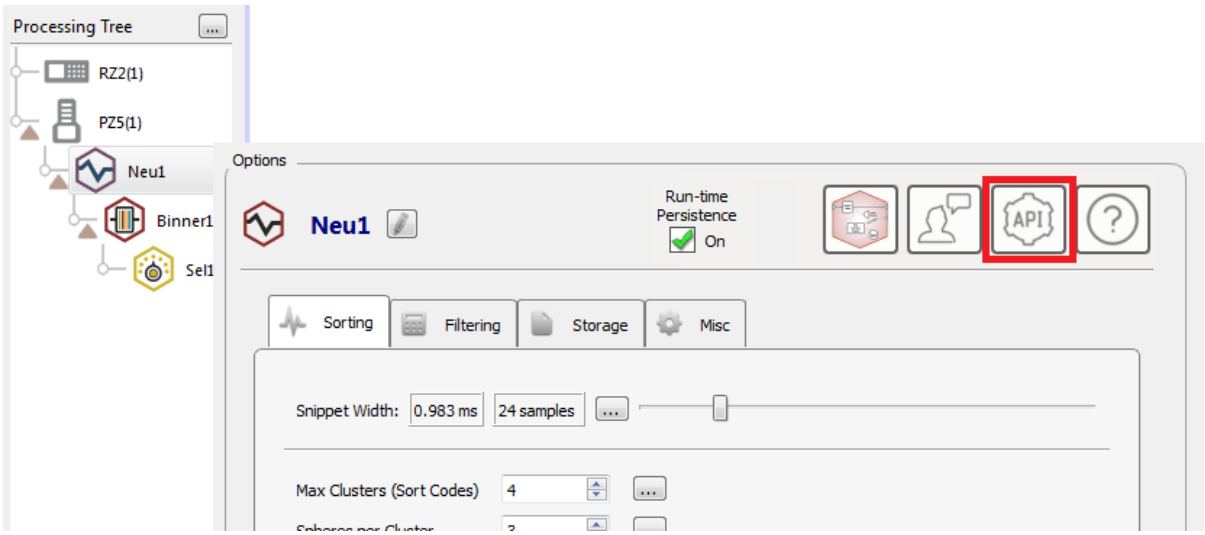
To enable SynapseAPI tools in the Synapse user interface, you must modify the Synapse preferences to start the built-in TCP Server.

1. In Synapse, click **Menu** and **Preferences**



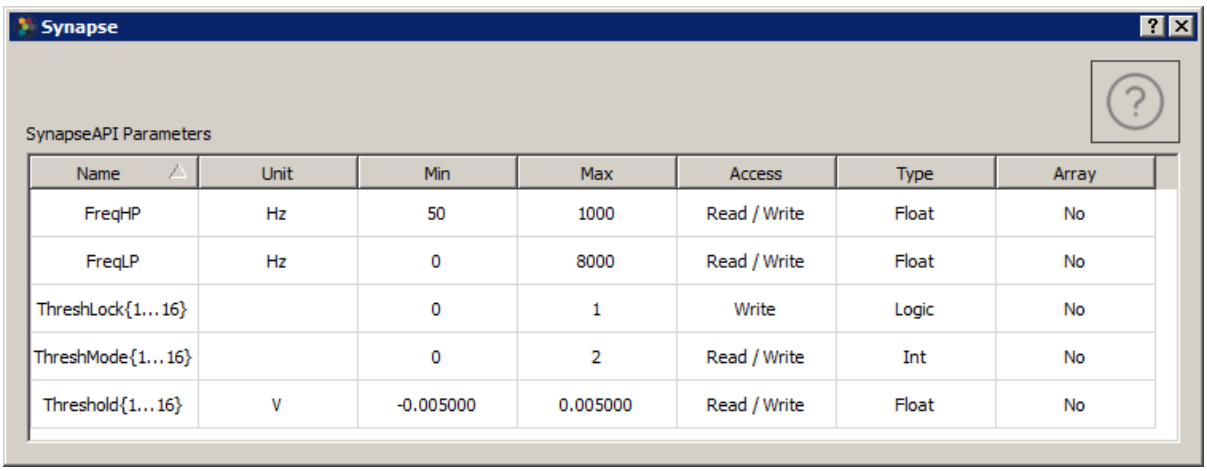
2. Click the **Synapse Server Enabled** check box
3. Click **OK**
4. Close and restart Synapse

Select a gizmo or hardware item in the Processing Tree, and if that item supports the API you will see an API button, pictured below, on the upper right of the Options page for that item.



PCA Spike Sporting Gizmo Options with SynapseAPI Tools Enabled

Clicking the API button displays the parameters available for the selected gizmo or hardware object.

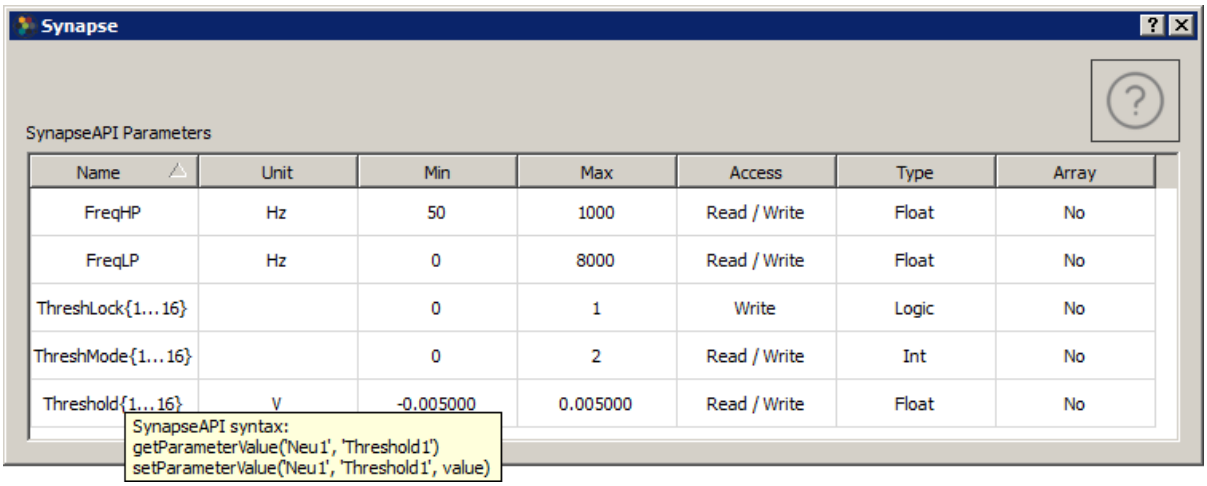


Name	Unit	Min	Max	Access	Type	Array
FreqHP	Hz	50	1000	Read / Write	Float	No
FreqLP	Hz	0	8000	Read / Write	Float	No
ThreshLock{1...16}		0	1	Write	Logic	No
ThreshMode{1...16}		0	2	Read / Write	Int	No
Threshold{1...16}	V	-0.005000	0.005000	Read / Write	Float	No

PCA Spike Sorting API Dialog

Each row describes the specified parameter with bounded values; including Min and Max values, the type of access allowed through the API, and the data type. The size of the parameter is also displayed if it contains more than one value. For example, you will see the size displayed for the Channel Mapper gizmo API.

The API dialog table is for information purposes only and does not allow you to modify the parameters. You can hover over the Name cell to view the syntax you would use in Python or MATLAB to get and set (if allowed) the value of the parameter.

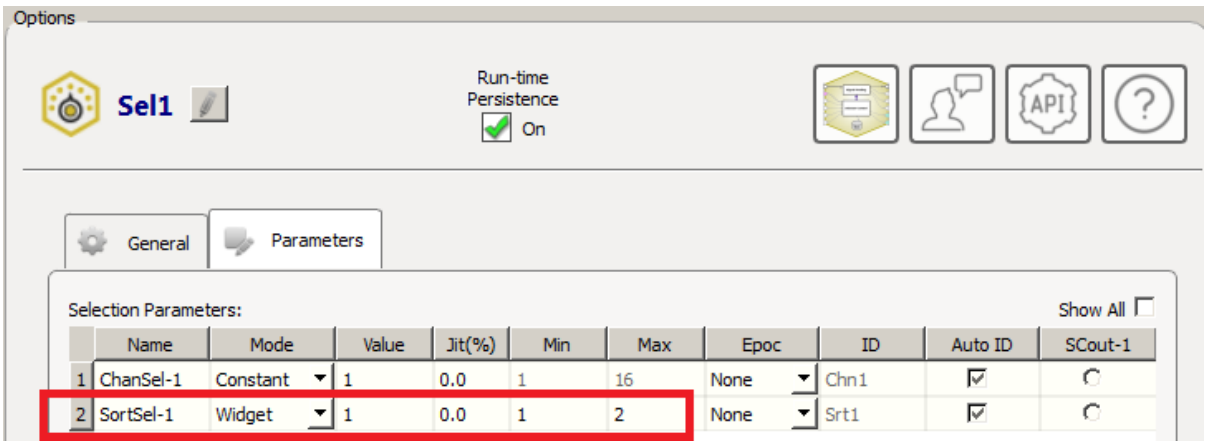


PCA Spike Sorting API Dialog with the Syntax Tooltip Displayed

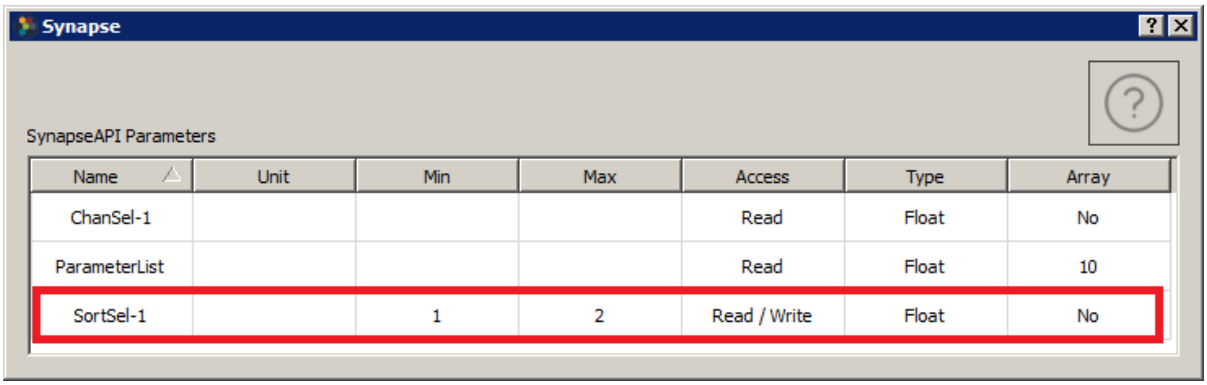
Parameters that have one entry per channel will condense into a single row for easier viewing. For example, the Threshold parameter of PCA Spike Sorting shown in the table above is actually 16 unique parameters, one for each channel in this example, called `Threshold1`, `Threshold2`, and so on up to `Threshold16`. You can't get/set all of the Thresholds at once, but instead you make separate calls to get/ set each one individually.

The parameters available for SynapseAPI access may be controlled during runtime only. For example, the table above displays the parameters available for the PCA Spike Sorting: high-pass and low-pass filter frequency and the spike detection threshold. When using SynapseAPI to set these values, you will see the widgets update in the Synapse runtime interface with the new values and these changes will also be logged into the Synapse database.

When accessing parameters defined in a bounded Parameters table, the desired parameter must be set to Widget Mode (runtime widget enabled), or else the parameter will not be available as a SynapseAPI parameter and cannot be modified at runtime. Below, the `SortSel-1` parameter of the Selector gizmo is set to Widget Mode.

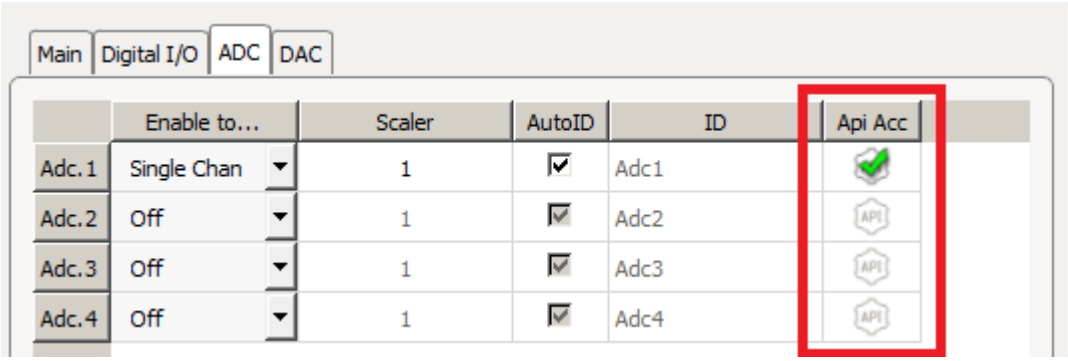


In this mode it then becomes a valid SynapseAPI parameter, and the API dialog shows the same Min/Max values defined in the Parameter Table.

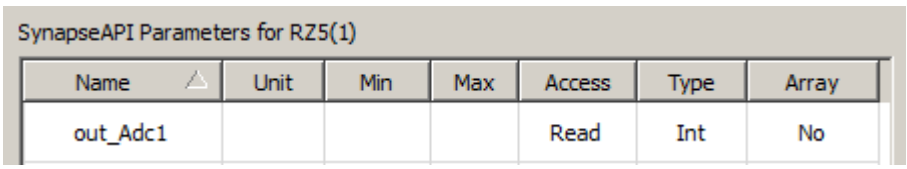


Directly Reading Signals

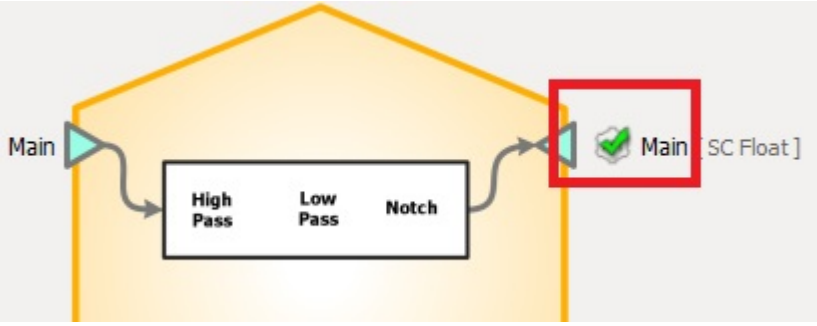
An API Access checkbox appears on any gizmo output link that can be read directly through the SynapseAPI. This includes some most gizmo output links and some RZ links. Some examples are below.



API Access for Adc1 on RZ



API dialog shows the parameter name to read Adc1



API Access for output of General Purpose Filter gizmo

SynapseAPI Parameters for Filt1

Name	Unit	Min	Max	Access	Type	Array
HighPassFreq	Hz	0.1	10000.0	Read / Write	Float	No
LowPassFreq	Hz	1.0	100000.0	Read / Write	Float	No
out_Main				Read	Float	No

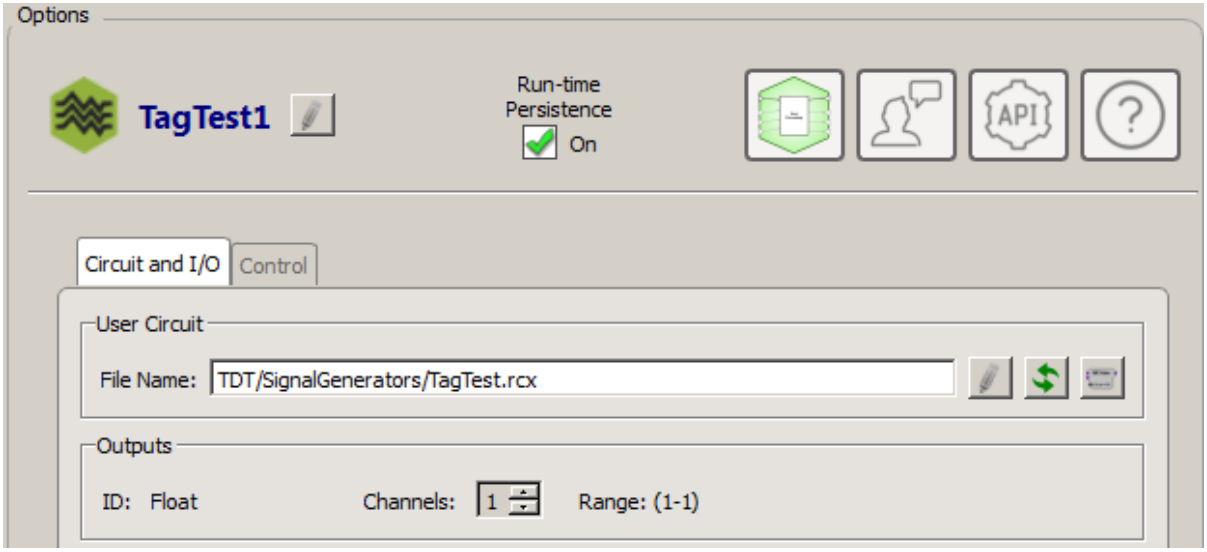
Updated API dialog shows the parameter name to read filter output

Interacting With User Gizmos

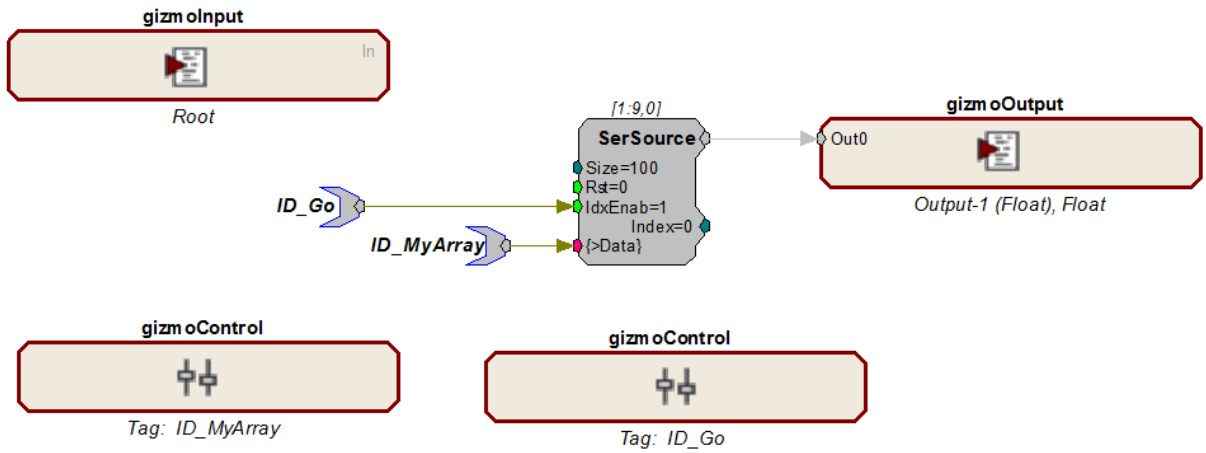
Any parameter in a User Gizmo that is defined by a matching parameter tag and gizmoControl macro pair is also accessible with SynapseAPI and will appear in the API dialog.

> **Note**
>
> See the User Gizmos section of the Synapse Manual for information on building and using User gizmos.

The example below shows the TagTest gizmo, which is available in the TDT > SignalGenerators category of Custom gizmos. This gizmo outputs an array of numbers controlled by the parameter called `MyArray` and turns the output on and off using a parameter called `Go`. Add the TagTest gizmo to your experiment, followed by a Stream Data Storage gizmo, to follow along with the example below.

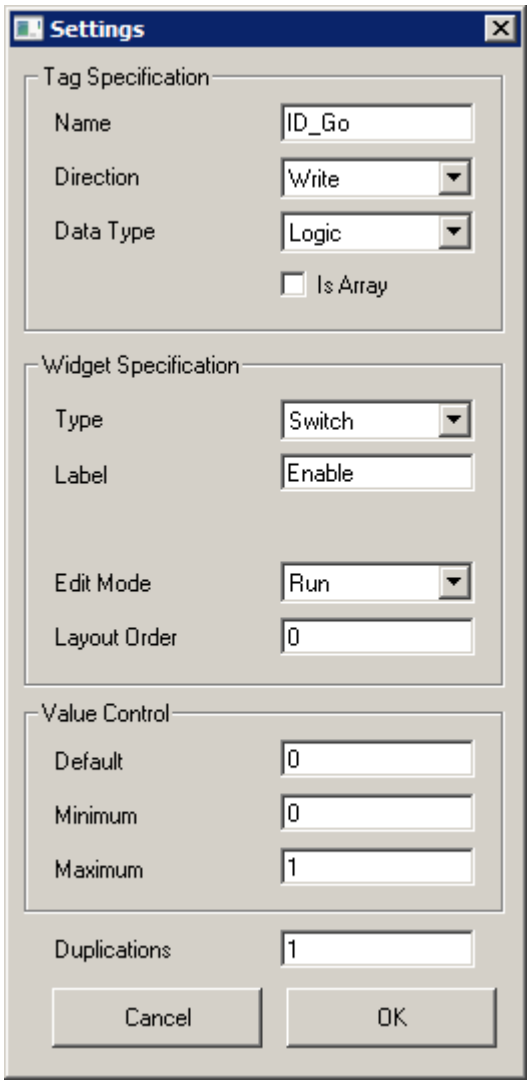


The RPvdsEx circuit for the gizmo (TagTest.rcx) contains four gizmo builder macros: gizmoInput, gizmoOutput, and two gizmoControls. The gizmoControl macros create the parameter specifications for this custom gizmo.

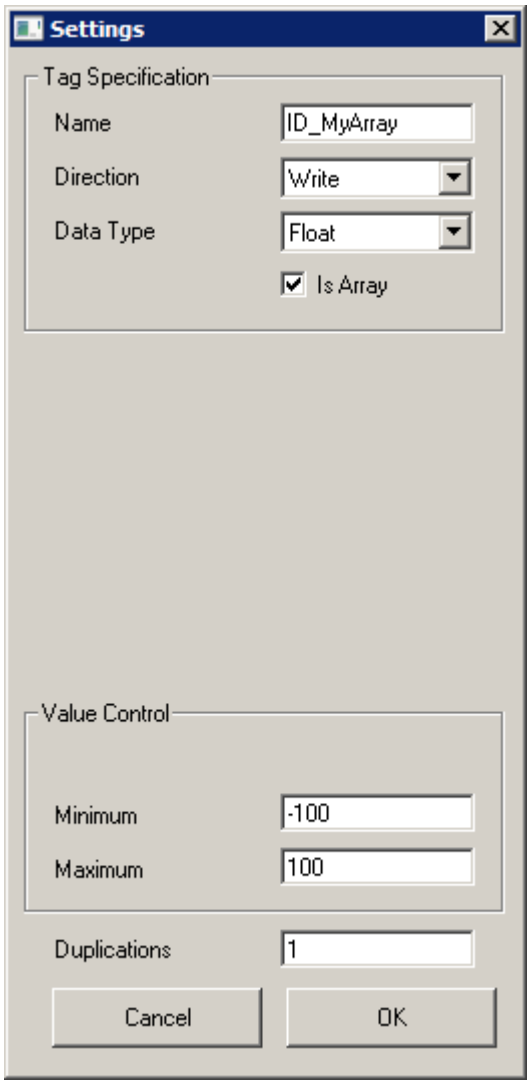


The first gizmoControl macro (`ID_Go`) turns the array test output on and off. By attaching the TagTest gizmo to a Stream Data Storage gizmo you can verify this at runtime.

In the macro settings for `ID_Go` , the parameter tag is defined as a logic value, and the runtime interface is a switch widget.

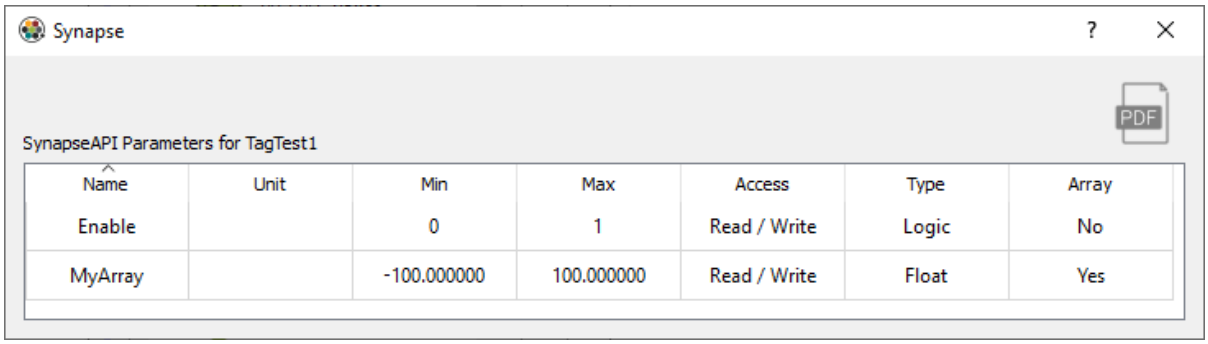


The circuit also includes a gizmoControl macro to define an array of values that can be written to dynamically and are sent as output when the test is running.



In the macro settings, the parameter tag is defined and the `Is Array` checkbox is selected. The bounds (`Minimum` and `Maximum`) for the parameter are applied to each element in the array whenever the API is writing the array values.

Because this user gizmo includes gizmoControl macros, the API button is shown in the Options area and the SynapseAPI Dialog can be displayed. Notice that the `MyArray` parameter is recognized as an Array. For User gizmos, the size of any parameter array can only be determined programmatically at runtime using getParameterSize, or by manually examining the circuit.



Name	Unit	Min	Max	Access	Type	Array
Enable		0	1	Read / Write	Logic	No
MyArray		-100.000000	100.000000	Read / Write	Float	Yes

Example

Interacting with the user gizmo parameters

MATLAB

```
% create SynapseAPI connection
syn = SynapseAPI('localhost');

% switch into a runtime mode (Preview in this case)
if syn.getMode() < 1, syn.setMode(2), end

% get all info on the 'MyArray' parameter
GIZMO = 'TagTest1';
PARAMETER = 'MyArray';
info = syn.getParameterInfo(GIZMO, PARAMETER)

% get the array size (should be 100)
sz = syn.getParameterSize(GIZMO, PARAMETER)

% write values 1 to 50 in second half of buffer
result = syn.setParameterValues(GIZMO, PARAMETER, 1:50, 50)

% read all values from buffer
syn.getParameterValues(GIZMO, PARAMETER, sz)

% get all info on the 'Go' parameter
PARAMETER = 'Go';
info = syn.getParameterInfo(GIZMO, PARAMETER)

% flip the switch
result = syn.setParameterValue(GIZMO, PARAMETER, 1)

% check the value
value = syn.getParameterValue(GIZMO, PARAMETER);
fprintf('value = %d\n', value);

% also verify visually that the switch slipped in the run
% time interface. This state change will be logged just
% like any other variable change and saved with the runtime
% state.
```

Python

```python
import numpy as np
import tdt

# create SynapseAPI connection
syn = tdt.SynapseAPI('localhost')

# switch into a runtime mode (Preview in this case)
if syn.getMode() < 1: syn.setMode(2)

# get all info on the 'MyArray' parameter
GIZMO = 'TagTest1'
PARAMETER = 'MyArray'
info = syn.getParameterInfo(GIZMO, PARAMETER)

# get the array size (should be 100)
sz = syn.getParameterSize(GIZMO, PARAMETER)

# write values 1 to 50 in second half of buffer
result = syn.setParameterValues(GIZMO, PARAMETER, np.arange(1, 51), 50)

# read all values from buffer
syn.getParameterValues(GIZMO, PARAMETER, sz)

# get all info on the 'Go' parameter
PARAMETER = 'Go'
info = syn.getParameterInfo(GIZMO, PARAMETER)

# flip the switch
result = syn.setParameterValue(GIZMO, PARAMETER, 1)

# check the value
value = syn.getParameterValue(GIZMO, PARAMETER)
print('value =', value)

# also verify visually that the switch slipped in the run
# time interface. This state change will be logged just
# like any other variable change and saved with the runtime
# state.
```

Recording Notes

The Run-time Notes feature built into the RZ and RX objects in Synapse lets you record timestamps and notes during a recording that are saved to disk.

You can assign common notes to buttons that are easily clicked by the user. There is also a text field to enter custom notes. The SynapseAPI gives you a way to record notes through your custom application.

Unlike other gizmos that use the API, the RZ and RX objects do not have the API button so their API parameters are not easily visible in the Synapse window.

The `Run-time Notes` mode must be set to `Notes File` or `Notes File + Epocs`. The API communicates with the 'RecordingNotes' tab at run-time during a Recording only.

Note

Note Capture does not work during Preview mode

Example

Adding Recording Notes using SynapseAPI

MATLAB

```
GIZMO = 'RecordingNotes';
BUTTON = 'Button';
NOTE = 'Note';

% log note associated with first button
syn.setParameterValue(GIZMO, BUTTON, 0)

% log note associated with second button
syn.setParameterValue(GIZMO, BUTTON, 1)

% write a custom note
syn.setParameterValue(GIZMO, NOTE, 'My Custom Note')
```

Python

```
GIZMO = 'RecordingNotes'
BUTTON = 'Button'
NOTE = 'Note'

# log note associated with first button
syn.setParameterValue(GIZMO, BUTTON, 0)

# log note associated with second button
syn.setParameterValue(GIZMO, BUTTON, 1)

# write a custom note
syn.setParameterValue(GIZMO, NOTE, 'My Custom Note')
```

API Reference

Mode Control

SynapseAPI

```
syn = SynapseAPI()
```

Returns an instance of SynapseAPI and connects to the Synapse server through port 24414. By default it connects to Synapse running on your local machine (localhost), but can optionally connect to Synapse running on a remote machine.

Example

Open a connection to the Synapse server.

MATLAB

```
% connect to locally running Synapse
syn = SynapseAPI();

% this is equivalent to SynapseAPI('localhost')
% or you can connect to a remote server on your network
syn = SynapseAPI('10.1.0.55');

% show the current system mode
syn.getModeStr()
```

Python

```
# connect to locally running Synapse
syn = SynapseAPI()

# this is equivalent to SynapseAPI('localhost')
# or you can connect to a remote server on your network
syn = SynapseAPI('10.1.0.55')

# show the current system mode
syn.getModeStr()
```

getMode

```
iMode = getMode()
```

Returns the current system mode of Synapse as an integer. This call can be used with setMode to control the operational mode of your entire system. The Synapse modes (Idle, Standby, Preview, and Record) are described in the Synapse Manual.

Returns	Type	Values
iMode	integer	0 (Idle), 1 (Standby), 2 (Preview), 3 (Record)

Example

Open a connection to the Synapse server, and if the mode is Record (3) then the routine runs.

MATLAB

```
syn = SynapseAPI();

if syn.getMode() == 3
    % Start Routine
end
```

Python

```
import tdt
syn = tdt.SynapseAPI()

if syn.getMode() == 3:
    # Start Routine
```

getModeStr

```
sMode = getMode()
```

Returns the current system mode of Synapse as a string. See description of getMode.

Returns	Type	Values
sMode	string	'Idle', 'Standby', 'Preview', 'Record'

Example

Open a connection to the Synapse server, and if the mode is 'Record' then the routine runs.

MATLAB

```
syn = SynapseAPI();

if strcmp(syn.getModeStr(), 'Record')
    % Start Routine
end
```

Python

```
import tdt
syn = tdt.SynapseAPI()

if syn.getModeStr() == 'Record':
    # Start Routine
```

setMode

```
bSuccess = setMode(iNewMode)
```

Sets the system mode of Synapse. The possible modes include: Idle, Standby, Preview, and Record.

Inputs	Type	Values
iNewMode	integer	0 (Idle), 1 (Standby), 2 (Preview), 3 (Record)
Returns	**Type**	**Values**
bSuccess	boolean	0 (fail), 1 (success)

Example

This code sample opens a connection to the Synapse server. If the Synapse mode is not Record mode (3), setMode puts Synapse in Record mode.

MATLAB

```
syn = SynapseAPI();

if syn.getMode() ~= 3
    syn.setMode(3);
end
```

Python

```
import tdt
syn = tdt.SynapseAPI()

if syn.getMode() != 3:
    syn.setMode(3)
```

setModeStr

```
bSuccess = setModeStr(iNewMode)
```

Sets the system mode of Synapse. The possible modes include: Idle, Standby, Preview, and Record. See description

Inputs	Type	Values
sNewMode	string	'Idle', 'Standby', 'Preview', 'Record'
Returns	**Type**	**Values**
bSuccess	boolean	0 (fail), 1 (success)

Example

This code sample opens a connection to the Synapse server. If the Synapse mode is not in 'Record' mode, setModeStr puts Synapse in 'Record' mode.

MATLAB

```
syn = SynapseAPI();

if strcmp(syn.setModeStr(), 'Record') ~= 1
    syn.setModeStr('Record');
end
```

Python

```
import tdt
syn = tdt.SynapseAPI()

if syn.getModeStr() != 'Record':
    syn.setModeStr('Record')
```

Gizmos and Parameters

getGizmoNames

```
cGizmos = getGizmoNames(bApiOnly)
```

Returns a cell array of all gizmos in the running experiment. This can be used with getParameterNames and get and set parameter methods to change parameters during runtime.

Inputs	Type	Values
bApiOnly	boolean	(optional) only return gizmos that have API parameters
Returns		
cGizmos	cell	strings of all gizmo names in current experiment

Example

Retrieve the gizmo names in current experiment.

MATLAB

```
gizmo_names = syn.getGizmoNames();

if numel(gizmo_names) < 1
    error('no gizmos found')
end

% return only the gizmos that have API parameters
gizmo_names = syn.getGizmoNames(1);
```

Python

```
gizmo_names = syn.getGizmoNames()

if len(gizmo_names) < 1:
    raise Exception('no gizmos found')

# return only the gizmos that have API parameters
gizmo_names = syn.getGizmoNames(1)
```

getGizmoParent

```
sGizmoParent = getGizmoParent(sGizmoName)
```

Return the name of the parent processor for a particular gizmo. Useful if you have multiple processors and want to know the sampling rate a particular gizmo is running at.

Inputs	Type	Values
sGizmoName	string	gizmo to find the parent of
Returns		
sGizmoParent	string	name of parent gizmo

Example

Get the gizmo's sampling rate by querying the parent processor.

MATLAB

```
parent = syn.getGizmoParent('aStim1');

rates = syn.getSamplingRates();

gizmoRate = rates.(parent);
```

Python

```
parent = syn.getGizmoParent('aStim1')

rates = syn.getSamplingRates()

gizmoRate = rates[parent]
```

getGizmoInfo

```
tGizmoInfo = getGizmoInfo(sGizmoName)
```

Returns a struct containing the gizmo type, category, description, and icon (string of base64-encoded text representing the icon).

Inputs	Type	Values
sGizmoName	string	gizmo name to retrieve information for
Returns		
tGizmoInfo	struct	struct fields: **cat** - gizmo category **desc** - gizmo description **icon** - base64-encoded icon **type** - gizmo type

Example

Find a specific type of gizmo in the current experiment

MATLAB

```
% get all gizmos
gizmos = syn.getGizmoNames();

% loop over gizmos, looking for 'AudioStim' gizmo
for gizmo = 1:numel(gizmos)
    info = syn.getGizmoInfo(gizmos{gizmo});
    if strcmp(info.type, 'AudioStim')
        disp('found AudioStim gizmo')
    end
end
```

Python

```
# get all gizmos
gizmos = syn.getGizmoNames()

# loop over gizmos, looking for 'AudioStim' gizmo
for gizmo in gizmos:
    info = syn.getGizmoInfo(gizmo)
    if strcmp(info.type, 'AudioStim'):
        print('found AudioStim gizmo')
```

getParameterNames

```
cParameters = getParameterNames(sGizmo)
```

Returns a cell array of parameter names for the specified gizmo. This can be used with getGizmoNames and get/set parameter methods to change parameters at runtime.

Inputs	Type	Values
sGizmo	string	gizmo to find the parent of
Returns		
cParameters	cell	strings of all parameter names for this gizmo

Example

Retrieve all parameter names of all gizmos

MATLAB

```
gizmo_names = syn.getGizmoNames()

for i = 1:numel(gizmo_names)
    gizmo = gizmo_names{i}
    params = syn.getParameterNames(gizmo);
end
```

Python

```
gizmo_names = syn.getGizmoNames()

for gizmo in gizmo_names:
    params = syn.getParameterNames(gizmo)
```

getParameterInfo

```
tParameterInfo = getParameterInfo(sGizmo, sParameter)
```

Returns a structure containing parameter information.

> **Note**
>
> The same information is displayed in a table in the Synapse designtime interface when you click the API button on the gizmo options tab.

Inputs	Type	Values
sGizmoName	string	gizmo name to retrieve information for
sParameter	string	parameter to retrieve information for
Returns		
tParameterInfo	struct	struct fields: **Unit** - the units label for this parameter **Min** - the minimum allowed value for this parameter **Max** - the maximum allowed value for this parameter **Type** - the parameters data type. Can be 'Float', 'Int', or 'Logic' **Array** - During designtime: If the parameter is an array of known size, this field contains the number of elements in this parameter. If the size is unknown (for example, User gizmo tag arrays or Legacy Hal parameter tags), this field contains 'Yes'. Otherwise it contains 'No' for any scalar parameters. During runtime: this field will always contain the size of the parameter if it is an array (>1), or 'No' if it is not.

Example

Retrieve all parameter info for all gizmos.

MATLAB

```
% get all gizmo names
gizmo_names = syn.getGizmoNames();

% loop through the names
for i = 1:numel(gizmo_names)
    gizmo = gizmo_names{i}

    % get all parameters for this gizmo
    params = syn.getParameterNames(gizmo)

    % loop through the parameters, get their info
    for i = 1:numel(params)
        param = params{i};
        info = syn.getParameterInfo(gizmo, param);
    end
end
```

Python

```
# get all gizmo names
gizmo_names = syn.getGizmoNames()

# loop through the names
for gizmo in gizmo_names:

    # get all parameters for this gizmo
    params = syn.getParameterNames(gizmo)

    # loop through the parameters, get their info
    for param in params:
        info = syn.getParameterInfo(gizmo, param)
```

getParameterSize

```
dValue = getParameterSize(sGizmo, sParameter)
```

Returns the size of the specified parameter from the specified gizmo. This can be used with getGizmoNames and getParameterNames.

Inputs	Type	Values
sGizmoName	string	gizmo name to retrieve information for
sParameter	string	parameter to retrieve information for
Returns		
dValue	double	size of parameter, will be > 1 if it is an array

Example

Retrieve all parameter sizes for all gizmos.

MATLAB

```
% get all gizmo names
gizmo_names = syn.getGizmoNames();

% loop through the names
for i = 1:numel(gizmo_names)
    gizmo = gizmo_names{i}

    % get all parameters for this gizmo
    params = syn.getParameterNames(gizmo)

    % loop through the parameters, get their sizes
    for i = 1:numel(params)
        param = params{i};
        param_size = syn.getParameterSize(gizmo);
    end
end
```

Python

```
# get all gizmo names
gizmo_names = syn.getGizmoNames()

# loop through the names
for gizmo in gizmo_names:

    # get all parameters for this gizmo
    params = syn.getParameterNames(gizmo)

    # loop through the parameters, get their sizes
    for param in params:
        param_size = syn.getParameterSize(gizmo, param)
```

getParameterValue

```
dValue = getParameterValue(sGizmo, sParameter)
```

Returns the value of the specified parameter from the specified gizmo. This can be used with getGizmoNames and getParameterNames.

Inputs	Type	Values
sGizmoName	string	gizmo name to retrieve value for
sParameter	string	parameter to retrieve value for
Returns		
dValue	double	current value of parameter

Example

Retrieve a General Purpose Filter gizmo's high pass frequency setting.

MATLAB

```
filter_highpass = syn.getParameterValue('Filt1', 'HighPassFreq')
```

Python

```
filter_highpass = syn.getParameterValue('Filt1', 'HighPassFreq')
```

setParameterValue

```
bSuccess = setParameterValue(sGizmo, sParameter, dValue)
```

Returns the value of the specified parameter from the specified gizmo. This can be used with getGizmoNames and getParameterNames.

Inputs	Type	Values
sGizmoName	string	gizmo name to set value for
sParameter	string	parameter to set value for
dValue	double	new value of parameter
Returns	**Type**	**Values**
bSuccess	boolean	0 (fail), 1 (success)

Example

Retrieve a General Purpose Filter high pass frequency, then increments it by 1.

MATLAB

```
val = syn.getParameterValue('Filt1', 'HighPassFreq')

syn.setParameterValue('Filt1', 'HighPassFreq', val + 1)

val = syn.getParameterValue('Filt1', 'HighPassFreq')
```

Python

```
val = syn.getParameterValue('Filt1', 'HighPassFreq')

syn.setParameterValue('Filt1', 'HighPassFreq', val + 1)

val = syn.getParameterValue('Filt1', 'HighPassFreq')
```

getParameterValues

```
fValues = getParameterValues(sGizmo, sParameter, count=-1, offset=0)
```

Returns the values of the specified parameter array.

Inputs	Type	Values
sGizmoName	string	gizmo name to retrieve value for
sParameter	string	parameter to retrieve value for
count	integer	(optional) number of values to return (default is to read the entire buffer)
offset	integer	(optional) buffer location to read from (default is beginning)
Returns		
fValues	float array	contents of the buffer

Example

Retrieve the map array from a Channel Mapper gizmo.

MATLAB

```
currMap = syn.getParameterValues('Map1', 'ChanMap')
```

Python

```
currMap = syn.getParameterValues('Map1', 'ChanMap')
```

setParameterValues

```
bSuccess = setParameterValue(sGizmo, sParameter, values, offset=0)
```

Returns the values of the specified parameter array.

Inputs	Type	Values
sGizmoName	string	gizmo name to retrieve value for
sParameter	string	parameter to retrieve value for
values	number	array of values to write into the parameter array
offset	integer	(optional) buffer location to write into (default is beginning)
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Set the map array of a Channel Mapper gizmo.

MATLAB

```
currMap = syn.getParameterValues('Map1', 'ChanMap')

defaultMap = 1:numel(currMap);

syn.setParameterValues('Map1', 'ChanMap', defaultMap);
```

Python

```
currMap = syn.getParameterValues('Map1', 'ChanMap')

defaultMap = 1:len(currMap)

syn.setParameterValues('Map1', 'ChanMap', defaultMap)
```

Lab Management

getKnownExperiments

```
cExperiments = getKnownExperiments()
```

Returns a cell array of the string values of known experiments.

Returns	Type	Values
cExperiments	cell	strings of all known experiment names

Example

Retrieve existing experiments from the Synapse database.

MATLAB

```
result = syn.getKnownExperiments()
if numel(result) < 1
    error('no experiments found')
end
```

Python

```
result = syn.getKnownExperiments()
if len(result) < 1:
    raise Exception('no experiments found')
```

getKnownSubjects

```
cSubjects = getKnownSubjects()
```

Returns a cell array of the string values of known subjects.

Returns	Type	Values
cSubjects	cell	strings of all known subject names

Example

Retrieve existing subjects from the Synapse database.

MATLAB

```
result = syn.getKnownSubjects()
if numel(result) < 1
    error('no subjects found')
end
```

Python

```
result = syn.getKnownSubjects()
if len(result) < 1:
    raise Exception('no subjects found')
```

getKnownUsers

```
cUsers = getKnownUsers()
```

Returns a cell array of the string values of known users.

Returns	Type	Values
cUsers	cell	strings of all known user names

Example

Retrieve existing users from the Synapse database.

MATLAB

```
result = syn.getKnownUsers()
if numel(result) < 1
    error('no users found')
end
```

Python

```
result = syn.getKnownUsers()
if len(result) < 1:
    raise Exception('no users found')
```

getKnownTanks

```
cTanks = getKnownTanks()
```

Returns a cell array of the string values of known tanks.

Returns	Type	Values
cTanks	cell	strings of all known tank names

Example

Retrieve existing tanks from from the Synapse database.

MATLAB

```
result = syn.getKnownTanks()
if numel(result) < 1
    error('no tanks found')
end
```

Python

```
result = syn.getKnownTanks()
if len(result) < 1:
    raise Exception('no tanks found')
```

getKnownBlocks

```
cBlocks = getKnownBlocks()
```

Returns a cell array of the string values of known blocks.

Returns	Type	Values
cBlocks	cell	strings of all known block names

Example

Retrieve existing block names from the Synapse database.

MATLAB

```
result = syn.getKnownBlocks()
if numel(result) < 1
    error('no blocks found')
end
```

Python

```
result = syn.getKnownBlocks()
if len(result) < 1:
    raise Exception('no blocks found')
```

getCurrentExperiment

```
sExperiment = getCurrentExperiment()
```

Returns the name of the currently selected experiment.

Returns	Type	Values
sExperiment	string	current experiment name

Example

Retrieve the current experiment name.

MATLAB

```
currExperiment = syn.getCurrentExperiment()
```

Python

```
currExperiment = syn.getCurrentExperiment()
```

getCurrentSubject

```
sSubject = getCurrentSubject()
```

Returns the name of the currently selected subject.

Returns	Type	Values
sSubject	string	current subject name

Example

Retrieve the current subject name.

MATLAB

```
currSubject = syn.getCurrentSubject()
```

Python

```
currSubject = syn.getCurrentSubject()
```

getCurrentUser

```
sUser = getCurrentUser()
```

Returns the name of the currently selected user.

Returns	Type	Values
sUser	string	current user name

Example

Retrieve the current user name.

MATLAB

```
currUser = syn.getCurrentUser()
```

Python

```
currUser = syn.getCurrentUser()
```

getCurrentTank

```
sTank = getCurrentTank()
```

Returns the name of the currently selected tank.

Returns	Type	Values
sTank	string	current tank name

Example

Retrieve the current tank name.

MATLAB

```
currTank = syn.getCurrentTank()
```

Python

```
currTank = syn.getCurrentTank()
```

getCurrentBlock

```
sBlock = getCurrentBlock()
```

Returns the name of the currently recording block.

Returns	Type	Values
sBlock	string	current block name

Example

Retrieve the current block name.

MATLAB

```
currBlock = syn.getCurrentBlock()
```

Python

```
currBlock = syn.getCurrentBlock()
```

createSubject

```
bSuccess = createSubject(sName, sDesc, sIcon)
```

Creates a subject with the given name, description, and icon.

Inputs	Type	Values
sName	string	name of subject
sDesc	string	description of subject
sIcon	string	type of icon to represent subject. Valid icons are: 'mouse', 'rat', 'monkey', 'marmoset', 'human', 'bat', 'owl', 'bird', 'ferret', 'gerbil', 'guinea-pig', 'rabbit', 'pig', 'cat', 'dog', 'fish', 'dolphin', 'snake', 'shark', 'duck', 'cow', 'goat', 'horse'
Returns		**Values**
bSuccess	boolean	0 (fail), 1 (success)

Example

Create a new subject, and set it as the current subject before recording.

MATLAB

```
nextSubj = 'ABC123';

syn.createSubject(nextSub, 'Control', 'mouse')

syn.setCurrentSubject(nextSub)

syn.setMode(3)
```

Python

```
nextSub = 'ABC123'

syn.createSubject(nextSub, 'Control', 'mouse')

syn.setCurrentSubject(nextSub)

syn.setMode(3)
```

createTank

```
bSuccess = createTank(sTankPath)
```

Creates a tank at the given path.

Inputs	Type	Values
sTankPath	string	path to new tank
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Create a new tank, set it as the current tank, and start recording.

MATLAB

```
nextTank = 'C:\TDT\NEXTTANK'

syn.createTank(nextTank)

syn.setCurrentTank(nextTank)

syn.setMode(3)
```

Python

```
nextTank = 'C:\\TDT\\NEXTTANK'

syn.createTank(nextTank)

syn.setCurrentTank(nextTank)

syn.setMode(3)
```

setCurrentExperiment

```
bSuccess = setCurrentExperiment(sExperiment)
```

Creates a tank at the given path.

Inputs	Type	Values
sExperiment	string	experiment name
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Find first known experiment and select it.

MATLAB

```
result = syn.knownExperiments()

syn.setCurrentExperiment(result{1})
```

Python

```
result = syn.knownExperiments()

syn.setCurrentExperiment(result[0])
```

setCurrentSubject

```
bSuccess = setCurrentSubject(sSubject)
```

Creates a tank at the given path.

Inputs	Type	Values
sSubject	string	subject name
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Find first known subject and select it.

MATLAB

```
result = syn.knownSubjects()

syn.setCurrentSubject(result{1})
```

Python

```
result = syn.knownSubjects()

syn.setCurrentSubject(result[0])
```

setCurrentUser

```
bSuccess = setCurrentUser(sUser, sPassword)
```

Creates a tank at the given path.

Inputs	Type	Values
sUser	string	user name
sPassword	string	(optional) user password, if set during creation
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Find first known user and select it.

MATLAB

```
result = syn.knownUsers()

syn.setCurrentUser(result{1})
```

Python

```
result = syn.knownUsers()

syn.setCurrentUser(result[0])
```

setCurrentTank

```
bSuccess = setCurrentTank(sTank)
```

Switches to the specified tank.

Note

The `Auto Tank Naming` option in the Preferences menu must be disabled for this to succeed.

Inputs	Type	Values
sTankPath	string	path to tank
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Set the tank name before the next recording.

MATLAB

```
nextTank = 'C:\TDT\SYNTANK';

syn.setMode(0)

syn.setCurrentTank(nextTank)

syn.setMode(3)

disp(syn.getCurrentTank())
```

Python

```
nextTank = 'C:\\TDT\\SYNTANK'

syn.setMode(0)

syn.setCurrentTank(nextTank)

syn.setMode(3)

print(syn.getCurrentTank())
```

setCurrentBlock

```
bSuccess = setCurrentBlock(sBlock)
```

Switches to the specified block.

Note

Note that the `Block Naming` option in the Preferences menu must be set to `Prompt` for this to succeed.

Inputs	Type	Values
sBlock	string	block name
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Set the block name before the next recording.

MATLAB

```
nextBlock = 'MyBlockName';

syn.setMode(0)

syn.setCurrentBlock(nextBlock)

syn.setMode(3)

disp(syn.getCurrentBlock())
```

Python

```
nextBlock = 'MyBlockName'

syn.setMode(0)

syn.setCurrentBlock(nextBlock)

syn.setMode(3)

print(syn.getCurrentBlock())
```

getSubjectMemos

```
cNotes = getSubjectMemos(sSubject)
```

Retrieves the log entries from the Synapse database for the specified subject. These can optionally be filtered by time using the STARTTIME and STOPTIME parameters which are formatted like %Y%m%d%H%M%S, e.g. May 22, 2020 9:53 am would be 20200522095300.

Inputs (MATLAB)	Type	Values
sSubject	string	subject name to search
STARTTIME	double	(optional) memo timestamp filter
STOPTIME	double	(optional) memo timestamp filter
Inputs (Python)		
sSubject	string	subject name to search
startTime	string	(optional) memo timestamp filter
stopTime	string	(optional) memo timestamp filter
memoNum	number	(optional) return a single memo (1-based index)
startNum	number	(optional) memo number filter (nonnegative)
endNum	number	(optional) memo number filter (nonnegative)
Returns		
cNotes	cell	array of filtered subject notes matching query parameters

Example

Retrieve specific notes from the Synapse database.

MATLAB

```
currSubject = syn.currentSubject()

allNotes = syn.getSubjectMemos(currSubject)

% get notes from 2019
filteredNotes = syn.getSubjectMemos(currSubject, 'STARTTIME',
    20190101000000, 'ENDTIME', 20191231235959)
```

Python

```
currSubject = syn.currentSubject()

allNotes = syn.getSubjectMemos(currSubject)

# get notes from 2019
filteredNotes = syn.getSubjectMemos(currSubject,
    startTime='20190101000000', endTime='20191231235959')
```

getExperimentMemos

```
cNotes = getExperimentMemos(sExperiment)
```

Retrieves the log entries from the Synapse database for the specified experiment. These can optionally be filtered by time using the STARTTIME and STOPTIME parameters which are formatted like %Y%m%d%H%M%S, e.g. May 22, 2020 9:53 am would be 20200522095300.

Inputs (MATLAB)	Type	Values
sExperiment	string	experiment name to search
STARTTIME	double	(optional) memo timestamp filter
STOPTIME	double	(optional) memo timestamp filter
Inputs (Python)		
sExperiment	string	experiment name to search
startTime	string	(optional) memo timestamp filter
stopTime	string	(optional) memo timestamp filter
memoNum	number	(optional) return a single memo (1-based index)
startNum	number	(optional) memo number filter (nonnegative)
endNum	number	(optional) memo number filter (nonnegative)
Returns		
cNotes	cell	array of filtered experiment notes matching query parameters

Example

Retrieve specific notes from the Synapse database.

MATLAB

```
currExperiment = syn.currentExperiment()

allNotes = syn.getExperimentMemos(currExperiment)

% get notes from 2019
filteredNotes = syn.getExperimentMemos(currExperiment, 'STARTTIME',
    20190101000000, 'ENDTIME', 20191231235959)
```

Python

```
currExperiment = syn.currentExperiment()

allNotes = syn.getExperimentMemos(currExperiment)

# get notes from 2019
filteredNotes = syn.getExperimentMemos(currExperiment,
    startTime='20190101000000', endTime='20191231235959')
```

getUserMemos

```
cNotes = getUserMemos(sUser)
```

Retrieves the log entries from the Synapse database for the specified user. These can optionally be filtered by time using the STARTTIME and STOPTIME parameters which are formatted like %Y%m%d%H%M%S, e.g. May 22, 2020 9:53 am would be 20200522095300.

Inputs (MATLAB)	Type	Values
sUser	string	user name to search
STARTTIME	double	(optional) memo timestamp filter
STOPTIME	double	(optional) memo timestamp filter
Inputs (Python)		
sUser	string	user name to search
startTime	string	(optional) memo timestamp filter
stopTime	string	(optional) memo timestamp filter
memoNum	number	(optional) return a single memo (1-based index)
startNum	number	(optional) memo number filter (nonnegative)
endNum	number	(optional) memo number filter (nonnegative)
Returns		
cNotes	cell	array of filtered user notes matching query parameters

Example

Retrieve specific notes from the Synapse database.

MATLAB

```
currUser = syn.currentUser()

allNotes = syn.getUserMemos(currUser)

% get notes from 2019
filteredNotes = syn.getUserMemos(currUser, 'STARTTIME',
    20190101000000, 'ENDTIME', 20191231235959)
```

Python

```
currUser = syn.currentUser()

allNotes = syn.getUserMemos(currUser)

# get notes from 2019
filteredNotes = syn.getUserMemos(currUser,
    startTime='20190101000000', endTime='20191231235959')
```

appendSubjectMemo

```
bSuccess = appendSubjectMemo(sSubject, sMemo)
```

Adds a database entry linked to this specified subject, just as if the user typed it in the Logs dialog.

Inputs	Type	Values
sSubject	string	name of subject
sMemo	string	memo to timestamp and add to subject
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Add memo to current subject.

MATLAB

```
currSubject = syn.currentSubject()

syn.appendSubjectMemo(currSubject, 'Subject memo from MATLAB')
```

Python

```
currSubject = syn.currentSubject()

syn.appendSubjectMemo(currSubject, 'Subject memo from Python')
```

appendExperimentMemo

```
bSuccess = appendExperimentMemo(sExperiment, sMemo)
```

Adds a database entry linked to this specified experiment, just as if the user typed it in the Logs dialog.

Inputs	Type	Values
sExperiment	string	name of experiment
sMemo	string	memo to timestamp and add to experiment
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Add memo to current experiment.

MATLAB

```
currExperiment = syn.currentExperiment()

syn.appendExperimentMemo(currExperiment, 'Experiment memo from MATLAB')
```

Python

```
currExperiment = syn.currentExperiment()

syn.appendExperimentMemo(currExperiment, 'Experiment memo from Python')
```

appendUserMemo

```
bSuccess = appendUserMemo(sUser, sMemo)
```

Adds a database entry linked to this specified user, just as if the user typed it in the Logs dialog.

Inputs	Type	Values
sUser	string	name of user
sMemo	string	memo to timestamp and add to user
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Add memo to current user.

MATLAB

```
currUser = syn.currentUser()

syn.appendUserMemo(currUser, 'User memo from MATLAB')
```

Python

```
currUser = syn.currentUser()

syn.appendUserMemo(currUser, 'User memo from Python')
```

Persistence

getPersistModes

```
cModes = getPersistModes()
```

Returns a cell array of the allowed persistence modes settable from the API.

Returns	Type	Values
cModes	cell	strings of allowed persistence modes. Can be 'Last', 'Best', or 'Fresh'

Example

Retrieve allowed persistence modes.

MATLAB

```
persist_modes = syn.getPersistModes()
```

Python

```
persist_modes = syn.getPersistModes()
```

getPersistMode

```
sMode = getPersistMode()
```

Returns the current persistence mode.

Returns	Type	Values
sMode	string	current persistence mode. Can be 'Last', 'Best', 'Fresh', or 'User'

Example

Retrieve allowed persistence modes.

MATLAB

```
persist_mode = syn.getPersistMode()
```

Python

```
persist_mode = syn.getPersistMode()
```

setPersistMode

```
bSuccess = setPersistMode(sMode)
```

Sets the current persistence mode.

Note

Synapse must be in Idle mode for this to work.

Inputs	Type	Values
sMode	string	persistence mode to change to. Can be 'Last', 'Best', or 'Fresh'
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Set persistence to 'Fresh' before entering run mode.

MATLAB

```
syn.setPersistMode('Fresh')

syn.setModeStr('Preview')
```

Python

```
syn.setPersistMode('Fresh')

syn.setModeStr('Preview')
```

System Status

getSystemStatus

```
tStatus = getSystemStatus()
```

Returns a structure containing system state information. This same information is found in the lower left corner of the Synapse main window.

Returns	Type	Values
tStatus	struct	struct with fields: **errorCount** - number of errors **recordSecs** - number of seconds recorded **rateMBps** - data rate as Mb per second **sysLoad** - IO load percentage **uiLoad** - UI load percentage

Example

Start a recording, wait 5 seconds, then retrieve system status information.

MATLAB

```
syn.setMode(3);
pause(5)
tStatus = syn.getSystemStatus();
```

Python

```
import time

syn.setMode(3)
time.sleep(5)
tStatus = syn.getSystemStatus()
```

getSamplingRates

```
tSamplingRates = getSamplingRates()
```

Returns a structure containing the sampling rates for each device in the Synapse Processing Tree.

Returns	Type	Values
tSamplingRates	struct	struct fields are the names of all devices

Example

Retrieve the device sampling rate for an RZ6 processor.

MATLAB

```
result = syn.getSamplingRates();
sf = result.RZ6_1;
```

Python

```
result = syn.getSamplingRates()
sf = result['RZ6(1)']
```

Miscellaneous Utilities

issueTrigger

```
bSuccess = issueTrigger(iTriggerId)
```

Fires the named software trigger to all devices. Used only by User Gizmos that contain a TrgIn component.

Inputs	Type	Values
iTriggerId	integer	software trigger id. Can be any integer 1-10
Returns		
bSuccess	boolean	0 (fail), 1 (success)

Example

Send software trigger 1 to all devices.

MATLAB

```
syn.issueTrigger(1)
```

Python

```
syn.issueTrigger(1)
```

getError

```
sError = getError()
```

Returns the previous error (if any) generated by a failure of one of the SynapseAPI functions.

Returns	Type	Values
sError	string	error message

Matlab Examples

MATLAB Examples

The MATLAB SDK includes the SynapseAPI class for communicating with Synapse.

You can download the latest MATLAB SDK files here. The SynapseAPI class installs into:

```
C:\TDT\TDTMatlabSDK\TDTSDK\SynapseAPI
```

Example experiments and scripts can be found here:

```
C:\TDT\TDTMatlabSDK\Examples\SynapseAPI
```

Run Experiment For Set Time

This examples shows how to use MATLAB to control the Synapse mode and monitor recording status

This uses the 'ExperimentSetTime' example experiment, but it could run with any experiment

Export Continuous Data To Corpus Emulator

Import continuous data into MATLAB using TDTbin2mat

Export to binary files that are read into Corpus as PZ5 data

Use SynapseAPI to generate new recordings with this data

Concatenate the resulting data sets into one master data structure

Trigger Input

This examples shows how to start a Synapse recording using an external trigger into your processor

MATLAB monitors the bit input and controls Synapse state

Run Experiment For Set Duration

This examples shows how to use MATLAB to control the Synapse mode and monitor recording status
This uses the 'ExperimentSetTime' example experiment, but it could run with any experiment

Download M File

Download Experiment File

Housekeeping

Clear workspace and close existing figures. Add SDK directories to MATLAB

```
close all; clear all; clc;
[APIEXAMPLESPATH,name,ext] = fileparts(cd); % \TDTMatlabSDK\Examples
[SDKPATH,name,ext] = fileparts(APIEXAMPLESPATH); % \TDTMatlabSDK
addpath(genpath(SDKPATH));
```

Setup

Choose which experiment to run and the duration. It could be anything. This example uses a simple experiment with just a Tick store. To see full list of available experiments use syn.getKnownExperiments()

```
EXPERIMENT = 'ExperimentSetTime';
TOTAL_TIME = 30;

% Connect to Synapse
syn = SynapseAPI();

% Set your experiment
syn.setCurrentExperiment(EXPERIMENT);
```

Runtime

Set the system to 'Preview' mode

```
syn.setModeStr('Preview');

% Wait five seconds to give 'getSystemStatus' time to update internally
pause(5);
```

Main Loop

```
currTime = 0; prevTime = 0;
% Poll the system status until it reaches the desired state
while currTime < TOTAL_TIME
    % Add any additional API controls here
    currTime = syn.getSystemStatus.iRecordSecs;
    if prevTime ~= currTime
        fprintf('Current elapsed time: %ds\n', currTime);
    end
    prevTime = currTime;
end
```

```
Current elapsed time: 1s
Current elapsed time: 2s
Current elapsed time: 3s
Current elapsed time: 4s
Current elapsed time: 5s
Current elapsed time: 6s
Current elapsed time: 7s
Current elapsed time: 8s
Current elapsed time: 9s
Current elapsed time: 10s
Current elapsed time: 11s
Current elapsed time: 12s
Current elapsed time: 13s
Current elapsed time: 14s
Current elapsed time: 15s
Current elapsed time: 16s
Current elapsed time: 17s
Current elapsed time: 18s
Current elapsed time: 19s
Current elapsed time: 20s
Current elapsed time: 21s
Current elapsed time: 22s
Current elapsed time: 23s
Current elapsed time: 24s
Current elapsed time: 25s
Current elapsed time: 26s
Current elapsed time: 27s
Current elapsed time: 28s
Current elapsed time: 29s
Current elapsed time: 30s
```

Our desired elapsed time has passed, switch to Idle mode

```
syn.setModeStr('Idle');
disp('done');
```

```
done
```

Runtime Output

Export Continuous Data To Corpus Emulator

Import continuous data into MATLAB using TDTbin2mat
Export to binary files that are read into Corpus as PZ5 data
Use SynapseAPI to generate new recordings with this data
Concatenate the resulting data sets into one master data structure

Download M File

Download Experiment File

Housekeeping

Clear workspace and close existing figures. Add SDK directories to MATLAB path.

```
close all; clear all; clc;
[MAINEXAMPLEPATH,name,ext] = fileparts(cd); % \TDTMatlabSDK\Examples
DATAPATH = fullfile(MAINEXAMPLEPATH, 'ExampleData'); %
\TDTMatlabSDK\Examples\ExampleData
[SDKPATH,name,ext] = fileparts(MAINEXAMPLEPATH); % \TDTMatlabSDK
addpath(genpath(SDKPATH));
```

Importing the Data

This example assumes you downloaded our example data sets (link) and extracted it into the \TDTMatlabSDK\Examples directory. To import your own data, replace `BLOCKPATH` with the path to your own data block.

In Synapse, you can find the block path in the database. Go to Menu > History. Find your block, then Right-Click > Copy path to clipboard.

```
BLOCKPATH = fullfile(DATAPATH,'Algernon-180308-130351');
```

Setup the variables for the data you want to extract

We will extract a stream store and output it to a Mat file

```
STORE = 'Wav1';
% the circuit loads and begins running before Synapse switches to record
% mode, so add 5 seconds of zeros to the front of it to make sure we are
% recording all of our data.
PADDING = 5;
```

Experiment setup

This example uses the *'Demo_TankReplayer' experiment included in the download. In the RZ HAL, PortC0 is enabled as an input and its epoc store is set to 'Full'. This is critical to capture when the PZ data is valid. This will create the PC0* epoc event that we can time filter on during extraction.

```
EXPERIMENT = '_Demo_TankReplayer';
TARGET_DEVICE = 'PZ5'; % PZ5 or PZ2
if ~strcmp(EXPERIMENT(1:6), '_Demo_')
    error('Experiment name must start with _Demo_ to automatically import
Corpus data')
end

% Connect to Synapse
syn = SynapseAPI();

% Set appropriate scale factor depending on our device type
if strcmpi(TARGET_DEVICE, 'PZ5')
    SCALE = 1e9 / 4;
elseif strcmpi(TARGET_DEVICE, 'PZ2') || strcmpi(TARGET_DEVICE, 'PZ3') ||
strcmpi(TARGET_DEVICE, 'PZ4')
    SCALE = 1e9;
end
```

Read the headers to see how much data we have

```
heads = TDTbin2mat(BLOCKPATH, 'TYPE', {'streams'}, 'STORE', STORE, 'HEADERS',
1);

% get the total data size first
nchan = double(max(heads.stores.(STORE).chan));
npts = double(heads.stores.(STORE).size - 10);

MAX_SAMPLES = 256e6 / 2; % Corpus can load up to 256M samples total. Cut it
in half to be safe.

% leave room for our digital trigger as well
nsamples = npts * double(numel(heads.stores.(STORE).chan)) * (nchan+1) /
nchan;
FS = heads.stores.(STORE).fs;
ZEROPAD = zeros(nchan, ceil(PADDING * FS));

% if it's too big, we need to split it up
iter = ceil((nsamples + numel(ZEROPAD)) / MAX_SAMPLES);

% figure out what T1 and T2 should be on each iteration
T1 = 0;
% subtract our pad off of it, plus a little extra to be safe
DELTA = MAX_SAMPLES / FS / (nchan + 1) - (PADDING + 0.5);

FOLDERNAME = EXPERIMENT(7:end);
ROOT = sprintf('C:\\TDT\\Corpus\\Simulation Files\\Synapse Demos\\%s',
FOLDERNAME);

blockPaths = cell(1,iter);
for ii = 1:iter
    % first set it to the default experiment
    syn.setCurrentExperiment('Experiment');
    refreshExperiment = tic;

    % read our data into MATLAB and create MAT file for PZ5 emulated data
    data = TDTbin2mat(BLOCKPATH, 'TYPE', {'streams'}, 'STORE', STORE,
'HEADERS', heads, 'T1', T1, 'T2', T1 + DELTA);
    folder = fullfile(ROOT, 'PZ1');
    mkdir(folder);
    OUTFILE = fullfile(folder, 'chan.mat');
    arr = [ZEROPAD data.streams.(STORE).data] * SCALE;

    % log it
    DURATION = size(arr, 2) / FS;
    fprintf('Wrote %s to output file %s\n', STORE, OUTFILE);
    fprintf('Sampling Rate:\t%.6f Hz\n', FS);
    fprintf('Num Channels:\t%d\n', size(data.streams.(STORE).data, 1));
    fprintf('Duration:\t\t%.1fs (includes padding)\n', DURATION);

    % make the digital input trigger that tells when the data is valid
```

```
    bits = [zeros(1, size(ZEROPAD, 2)) ones(1, size(data.streams.(STORE).data,
2))];

    clear data;
    save(OUTFILE, 'arr');
    clear arr;

    % save the digital input trigger data into Corpus directory
    folder = fullfile(ROOT, 'RZ1');
    mkdir(folder);
    OUTFILE = fullfile(folder, 'bits.i16');
    fileID = fopen(OUTFILE,'w');
    fwrite(fileID, bits, 'int16');
    fclose(fileID);
    fprintf('Wrote Sync Bit0 to output file %s\n', OUTFILE);
    clear bits;

    % Reload your experiment to force Corpus to reload the new data
    % then to our experiment

    % make sure enough time has elapsed that Corpus detected the change
    while tic - refreshExperiment < 3
        pause(.1);
    end

    % load our experiment so Corpus catches it
    syn.setCurrentExperiment(EXPERIMENT);

    disp('Wait until data has loaded into Corpus (estimated)')
    pause(max(DURATION/10, 5))
    syn.setModeStr('Record');

    fprintf('Wait until data has completely cycled through (%d seconds)\n',
ceil(DURATION))
    pause(PADDING);
    while syn.getSystemStatus.iRecordSecs < ceil(DURATION)
        pause(1)
    end

    % store this block name
    thisBlockPath = fullfile(syn.getCurrentTank, syn.getCurrentBlock);
    fprintf('Done recording into: %s\n', thisBlockPath)
    blockPaths{ii} = thisBlockPath;

    % get ready for next iteration
    syn.setModeStr('Idle');
    T1 = T1 + DELTA;
end
```

```
read from t=0.00s to t=61.23s
Warning: Directory already exists.
Wrote Wav1 to output file C:\TDT\Corpus\Simulation Files\Synapse
Demos\TankReplayer\PZ1\chan.mat
Sampling Rate:  24414.062500 Hz
Num Channels:   16
Duration:       66.1s (includes padding)
Warning: Directory already exists.
Wrote Sync Bit0 to output file C:\TDT\Corpus\Simulation Files\Synapse
Demos\TankReplayer\RZ1\bits.i16
Wait until data has loaded into Corpus (estimated)
Wait until data has completely cycled through (67 seconds)
Done recording into: F:
\Tanks\_Demo_TankReplayer-220120-164017\Subject1-220120-164017
```

Import the new data back into MATLAB

```
disp('Reading the data back into MATLAB')
for ii = 1:numel(blockPaths)
    tempdata = TDTbin2mat(blockPaths{ii}, 'TYPE', {'epocs'}); % get just the
epoc events

    % we just want the first valid epoc event
    tr = [tempdata.epocs.PC0_.onset(1); tempdata.epocs.PC0_.offset(1)];

    % extract our data, just on that time range
    tempdata = TDTbin2mat(blockPaths{ii}, 'RANGES', tr);

    % your custom analysis code here
end
```

```
Reading the data back into MATLAB
read from t=0.00s to t=67.66s
read from t=0.00s to t=67.66s
```

Runtime Output

Trigger Input

This examples shows how to start a Synapse recording using an external trigger into your processor
MATLAB monitors the bit input and controls Synapse state

Download M File

Download Experiment File

Housekeeping

Clear workspace and close existing figures. Add SDK directories to MATLAB

```
close all; clear all; clc;
[APIEXAMPLESPATH,name,ext] = fileparts(cd); % \TDTMatlabSDK\Examples
[SDKPATH,name,ext] = fileparts(APIEXAMPLESPATH); % \TDTMatlabSDK
addpath(genpath(SDKPATH));
```

Setup

This example uses the 'TrigIn' experiment included in the download.

```
syn = SynapseAPI();
EXPERIMENT = 'TrigIn';
syn.setCurrentExperiment(EXPERIMENT);
```

Runtime

Move into Standby Mode to wait for external trigger. Note: You must have Standby Mode enabled in Menu > Preferences.

```
syn.setMode(1);
```

Wait for external trigger to move to Preview mode

```
prevTime = 0; tic;
while syn.getParameterValue('Trigger1','TrigIn') == 0
    currTime = floor(toc);
    if currTime > prevTime
        fprintf('Waiting %d seconds for trigger\n', currTime);
    end
    prevTime = currTime;
end
fprintf('%.1f seconds elapsed.\nGo!\n', toc);
syn.setModeStr('Preview');

% Experiment Loop
pause(5);

% Move from Preview to Idle Mode when done
syn.setModeStr('Idle');
```

```
Waiting 1 seconds for trigger
Waiting 2 seconds for trigger
Waiting 3 seconds for trigger
Waiting 4 seconds for trigger
Waiting 5 seconds for trigger
5.2 seconds elapsed.
Go!
```

Runtime Output

Python Notebooks

Example Python Notebooks for SynapseAPI

Experiment Set Time

Experiment Set Time

This examples shows how to use Python to control the Synapse mode and monitor recording status. This uses the 'ExperimentSetTime' example experiment, but it could run with any experiment.

Download Experiment
Download Notebook
Download as Python file

Run Experiment For Set Duration

This examples shows how to use Python to control the Synapse mode and monitor recording status.
This uses the 'ExperimentSetTime' example experiment, but it could run with any experiment.

Housekeeping

Import tdt library for SynapseAPI, and Python's built-in time library so we can add a pause

```
import time
import tdt
```

Setup

Choose which experiment to run and the duration. It could be anything.
This example uses a simple experiment with just a Tick store.
To see full list of available experiments use syn.getKnownExperiments()

```
EXPERIMENT = 'ExperimentSetTime'
TOTAL_TIME = 30

# Connect to Synapse
syn = tdt.SynapseAPI()

# Set your experiment
syn.setCurrentExperiment(EXPERIMENT)
```

Runtime

```
# Set the system to 'Preview' mode
syn.setModeStr('Preview')

# Wait five seconds to give 'getSystemStatus' time to update internally
time.sleep(5)
```

Main Loop

```
currTime = 0
prevTime = 0

# Poll the system status until it reaches the desired state
while currTime < TOTAL_TIME:

    # Add any additional API controls here
    currTime = syn.getSystemStatus()['recordSecs']
    if prevTime != currTime:
        print(f'Current elapsed time: {currTime}s')

    prevTime = currTime
```

```
Current elapsed time: 3s
Current elapsed time: 4s
Current elapsed time: 5s
Current elapsed time: 6s
Current elapsed time: 7s
Current elapsed time: 8s
Current elapsed time: 9s
Current elapsed time: 10s
Current elapsed time: 11s
Current elapsed time: 12s
Current elapsed time: 13s
Current elapsed time: 14s
Current elapsed time: 15s
Current elapsed time: 16s
Current elapsed time: 17s
Current elapsed time: 18s
Current elapsed time: 19s
Current elapsed time: 20s
Current elapsed time: 21s
Current elapsed time: 22s
Current elapsed time: 23s
Current elapsed time: 24s
Current elapsed time: 25s
Current elapsed time: 26s
Current elapsed time: 27s
Current elapsed time: 28s
Current elapsed time: 29s
Current elapsed time: 30s
```

Our desired elapsed time has passed, switch to Idle mode

```
syn.setModeStr('Idle')
print('done')
```

```
done
```