Headless mode

TSG's "headless" or unsupervised mode is about running TSG without any windows. TSG runs invisibly in the background, taking instruction from a script. You can't script TSG to do *everything* it normally does though. It understands only a few specific tasks in headless mode, e.g., download from database, update dataset, run copy-processing, run downsampling. A log file is created alongside the script file.[1] The log file records what was done and is worth a look.
We introduced headless mode some years ago so that we could offer scheduled database upload & download, and have TSG provide the business end of the NVCL TSG-dataset download service. As it was controlled by a scheduling dialog in TSG or behind the scenes by NVCL software, its workings were unknown to most TSG users. However it has now been expanded with new tasks. It still has a HyLogging focus but is starting to include support for a more general production-oriented workflow.

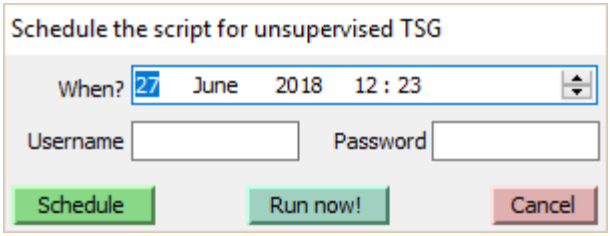
This document starts off with some information about how to schedule a headless script. (A script is scheduled, not run live.) Drag & drop is your friend here. After that it grinds through the script format[2]. To begin with, you'll find out about the overall script format, then two "global" script things: the strange and avoidable "batchloop", and the magnificent "multifile"[3]. Then we come to the main items – the tasks themselves. If there's anything I (the TSG programmer) want TSG to be able to do in Headless mode, I have to code a task for it. Currently there are only a few tasks but I add a new one every so often.
The quick links below skip past the general information to specific tasks.

Quick task links

[1] If this isn't the first time the script was run and a log file is there already, another log file is created. The log files are numbered.
[2] The scripts used here are completely different to TSG "batch scripts" that you might have encountered when creating "batch script" or "specialist" scalars.
[3] Batchloop is for repeating a whole script and I don't expect many people to use it. Multifile is very useful though. It's about wildcard file matching – *real* batch processing – so you should get your head around it.

Invocation

The easiest way to invoke headless mode is via **File -> Special -> Schedule a script for unsupervised TSG**. You will be prompted to select a script file, then you will get a scheduler dialog like the one below. (*Alternatively, just drag & drop a script file onto TSG and you will be taken straight to this dialog*.) Set the time that you'd like it to be run and type in your Windows login's username and password. (Note these credentials are not remembered by TSG.) Click **Schedule** and you'll get a message box telling you if the job was accepted or not. After that you can exit TSG and even log off. Just leave your computer turned on.



Update A **Run now!** Button has been added (June 2018). Click it and the script will be run straight away by a new, headless instance of TSG. You do not have to provide a time, username or password for this. You can exit the TSG that you are using but don't log off or switch off your computer until the run has completed.

For the brave at heart

More conventionally, headless mode is invoked by running TSG with the command-line switch **`/script=scriptfile`**.

Say there's a script-file `c:\mystuff\testscript.txt` and the TSG executable is installed as `C:\Users\fixthis\AppData\Local\The Spectral Geologist\tsgeol8.exe`.

1. Click the Windows "Start" button and begin typing the text "command prompt" in the text box labelled "Search programs and files". Select the tool when Windows finds it. Alternatively, you should find the command-prompt tool in Start -> Applications. Either way, you now have a DOS command prompt. (Handle it carefully because it could make your hair go grey.)
2. Type in the following line (*one line*) at the command prompt, after adjusting the path to work for you:
   `"C:\Users\fixthis\AppData\Local\The Spectral Geologist\tsgeol8.exe" /script=c:\mystuff\testscript.txt` Mind the quotes. If that line doesn't work then you probably don't have the correct TSG path. Right-click your TSG desktop icon and look at "properties" to find out where TSG is actually installed.

Script file

A TSG headless-mode script file is a plain text file. Here's an example:

```
multioptions vswir nodive nofield
multifile c:\00me\swift\0t\;  c:\00me\swift\topical\ddh*.tsg

task_begin
operation copyproc
update_items all
cproc_options scalars layout
cproc_template C:\00me\swift\seddy\14993_DD88MS1_Mt_Sedgwick_tsg.tsg
task_end

task_begin
operation downsamp
downsamp_config C:\00me\swift\downsampj50.cfg
output_dir c:\00me\0ds
dstolerate scalar
task_end
```

Here's some idle chatter before we get stuck in...

There's no special header at the start to identify the file as a "headless TSG script file" but of course the file is full of special tokens and things. Blank lines are ignored. A line starting with "#" or ";"is treated as a comment and ignored. A long line must not be split with a (typed in) carriage-return. Case is ignored.
Remember that this is TSG, which is... mature, so you must (as always) give it old-fashioned ASCII (7-bit ANSI) text files and not Unicode ones. It can be almost impossible to tell the difference just by looking in a text editor but there are clues. A Unicode text file uses about twice the disk space of an ASCII one, and a smart text editor like Notepad++ will show you what you have if you know where to look. If you have a script file that TSG chokes on then it might be Unicode. Open it in Notepad++ and look in the "Encoding" menu. It ought to have "Encode in ANSI" selected. If it doesn't, select "Convert to ANSI" and save.

Overall format

```
BATCHLOOP
MULTI_SPEC
TASK_SPEC
TASK_SPEC
...
TASK_SPEC
```

BATCHLOOP and MULTI_SPEC are both optional.
There must be at least one TASK_SPEC. Insofar as it *can* be common at this time, it is common to have just one.

BATCHLOOP

`BATCHLOOP` *seconds,loops*

The BATCHLOOP line is optional and if given it must be the first line in the script. It can be used to run (all tasks in) the script over and over. There might be occasions when it is useful.
It is given with two integer parameters, separated by a comma. The first is a wait time in seconds between loops, and the second is the number of loops (0 for infinite).
E.g., **BATCHLOOP 3600,12** will run the script 12 times with a pause of 1 hour between each run.
Looping is cancelled if a script task returns an error status.

MULTI_SPEC

`MULTIOPTIONS` *space-delimited options*
`MULTIFILE` *semicolon-delimited items*

The two MULTI_SPEC lines are optional. If given they must be at the start of the script (or immediately after `BATCHLOOP` if that is given). If a multi_spec is given in the body of a script then it is ignored.
This system is used to get the script's tasks to run on several TSG datasets (one at a time). It's new (in a relative sense). In the past, each task (except the database-download task) required a `TSG_DATASET` line specifying the dataset to use. That mechanism is still in place but is *overridden* by the MULTI_SPEC system (if MULTI_SPEC is used).

MULTIFILE

MULTIFILE has one or more items. If there's more than one then use semi-colons to delimit them.
In the future I might open up what an "item" can be, but currently it's:

- a path\filename,
- a path\filename with bits of the filename"wild-carded", or
- a path (preferably ending with '\').

If you know exactly which TSG datasets you want to batch-process then you can enumerate all of their filenames here (a rather long line) if you are so inclined. Most of us would rather use wildcard specs or whole directories, along with MULTIOPTIONS to restrict matches.
Last notes on MULTIFILE:

- Provide full paths that start from a drive letter (e.g., `d:\`) or a network drive spec (e.g., `\\clay-nm\pete\`). Headless TSG probably won't understand relative paths.
- There's no need to restrict wildcard matching to *.tsg. Feel free to use *.* Presently the headless system only acts on .TSG files that are found.
- `path\*.*` is equivalent to `path\`

MULTIOPTIONS

MULTIOPTIONS is mostly used to restrict the multifile wildcard search and to moderate the files passed on from the search for processing, but it can moderate all multi-spec activity[4]. It takes one or more space-delimited keywords:

- `NOSEARCH` Do not "find" anything; pass on the MULTIFILE items as they are. (There might be a use for this one day. Presently there isn't.)
- `NODIVE` Do not search down subdirectories.
- `VSWIR` Do not accept any TIR (thermal) datasets that are found. (`SWIR` and `VNIR` are equivalent keywords.)
- `TIR` Only accept TIR datasets.
- **`NOASSOC`** Instruct tasks not to open primary+associated. They must just open the single dataset they're given. (Not all tasks care about this.)
  Note: `NOASSOC` even affects task behaviour when the MULTIFILE system isn't used.
- `NOCORE` Do not accept a dataset that has the basic core-logging scalars.
- `NOFIELD` Do not accept a dataset that doesn't have the basic core-logging scalars. (Maybe you can exclude AUX datasets like this.)
- `TESTROCKS` Only find TestRocks datasets.
- `DIRS` This is a **new option** (November 2018). It causes the multifile wildcard search to look for directories rather than files. It turns on `NODIVE` implicitly, restricting the search to the level immediately below the given directory. Currently it is only used by the CLIMPORT task.

Keep in mind that `NOASSOC` is a special, general option.
Think about what files you have and what you want to do with them. For example, if you have a bunch of VSWIR+TIR datasets and you want to run an UPDATE task on all of them, my recommendation is to leave `NOASSOC` unmentioned (i.e., assoc open left active) and include the `VSWIR` option to stop the TIR datasets from being "found" individually. The UPDATE task knows how to open primary+assoc dataset pairs. Given a primary VSWIR dataset, it will also open (and update) the associated TIR dataset (provided there is one). If you did *not* include the MULTIOPTIONS `VSWIR` filter, you'd end up putting everything through twice. A dataset pair would be opened as VSWIR+TIR when the VSWIR file was "found", and again as TIR+VSWIR when the TIR file was "found".

TASK_SPEC

```
TASK_BEGIN
OPERATION task-name
Task-specific lines
TASK_END
```

That's the generic form of a task spec. Their contents vary. Optional task parameters are shown in **blue** in this document, and there are **default** settings for some of these.

---

[4] For example, if you enumerate a dozen individual TSG files (no wildcards) in MULTIFILE and use `MULTIOPTIONS VSWIR` then any TIR files amongst your dozen will be rejected.

UPLOAD

```
TASK_BEGIN
OPERATION UPLOAD
CONNECTION_STRING database URL
USERNAME database login
PASSWORD database login
DATABASE_TYPE currently ORACLE (the usual system default) or SQLSERVER
DUTOLERATE space-delimited options
TSG_DATASET path+filename of .TSG file (overridden by MULTI_SPEC)
RENAME override for stored user-friendly dataset name
HOLENAME override for stored drillhole name
HYLOGGER override for stored HyLogger name
PUBLISHED YES or NO
LINESCAN upload linescan raster attachment again – YES or NO
UPDATEOLDFMT automatically update old-format TSG datasets (just the dataset version
not the scalars) – YES or NO
FORCESCALARS Force the upload of scalars regardless of date – NONE, TSA or ALL
TASK_END
```

This is the NVCL database upload task. It currently honours the NOASSOC option so wotchit. NOASSOC is probably something that you don't normally want to use here.

- The DATABASE_TYPE line is optional.
- If MULTI_SPEC is in use then all lines from DUTOLERATE onwards are optional and in fact any TSG_DATASET, RENAME and HOLENAME lines are ignored. If MULTI_SPEC is not in use then **TSG_DATASET** must be given. It is recommended that you consider taking control of **PUBLISHED** (otherwise the dataset's "published" setting will be used).
- **DUTOLERATE** is optional. If not given then TSG won't tolerate any of the problems listed below. If given, it takes one or more space-separated keywords that relax certain aspects of quality control:
  - **LSLOST** Allow the upload of a dataset that currently has no attached linescan raster but whose internal tracking shows that it once did.
  - **PROFLOST** Allow the upload of a dataset that currently has no attached profilometer file but whose internal tracking shows that it once did.
  - **LSBAD** Allow the upload of a dataset whose attached linescan raster's internal tray / section metadata tables are faulty and unusable.
  - **TRAYPICS** Allow the upload of a dataset that doesn't have accompanying tray pictures.
  - **WFS** Allow the upload of a dataset that doesn't have a recorded borehole WFS for identifying its stuff in the elusive borehole database.
  - **UTSA** Allow the upload of a dataset that doesn't have user-level TSA scalars. (For a dataset pair, *both* datasets ought to have user TSA scalars.)
- **LINESCAN** is optional, and only meaningful if the dataset was uploaded before. In the past, a dataset's linescan was only uploaded the first time. It was left out of subsequent uploads of the modified dataset. Now you can include the linescan in a subsequent upload with LINESCAN YES.

Example

```
task_begin
operation upload
connection_string oratest1-cdc.it.csiro.au:1534/HYLOGNR.it.csiro.au
username yournamehere
password asifiddothat
```

dutolerate traypic wfs utsa
tsg_dataset C:\00me\swift\BEU162 HyLogger-1\BEU-162_tsg.tsg
rename BEU-162
holename beu-162
hylogger NA or Unknown
task_end

DOWNLOAD

TASK_BEGIN
OPERATION **DOWNLOAD**
CONNECTION_STRING *database URL*
USERNAME *database login*
PASSWORD *database login*
DATABASE_TYPE *currently ORACLE (the usual system default) or SQLSERVER*
OUTPUT_DIR *directory where the TSG dataset is to end up*
UUID *unique ID of dataset to download*
MATCH_STRING *all or part of the user-friendly name of the dataset to download*
SPECTRA *download spectra – YES or NO (NO gives a scalars-only dataset)*
LINESCAN *download linescan raster attachment – YES or NO*
PROFILOMETER *download profilometer attachment – YES or NO*
TRAYPICS *download tray pictures – YES or NO*
MOSPIC *download mosaic picture – YES or NO*
MAPPICS *download map pictures – YES or NO*
TASK_END

This is the NVCL database download task. It is fiercely independent – it ignores the NOASSOC option and currently doesn't take part in the MULTI_SPEC system.

- The DATABASE_**TYPE** line is optional.
- A path must be given in **OUTPUT_DIR**. Subdirectories will be created as required.
- Either **UUID** or MATCH_STRING must be given to identify the dataset to be downloaded. (Giving both is unacceptable.) UUID is better as it identifies the dataset unambiguously. If MATCH_STRING is given then the database is searched on the user-friendly dataset name field (which might not be unique) and the first match is taken.
- The remaining options (SPECTRA, LINESCAN, PROFILOMETER, TRAYPICS, MOSPIC, MAPPICS) are for controlling which dataset components are downloaded. They are all set to YES (on) by default.

Example

Task_begin
Operation download
Connection_string ncrisdb1-mi.vm.csiro.au/hylognr.ncrisdb1-mi.vm.csiro.au
Database_type oracle
Username meagain
Password stillnotgivingit
Uuid 6dd70215-fe38-457c-be42-3b165fd98c7
Output_dir C:\00me\swift\dataout
Task_end

UPDATE

```
TASK_BEGIN
OPERATION UPDATE
TSG_DATASET  path+filename of .TSG file (overridden by MULTI_SPEC)
UPDATE_ITEMS  space-delimited options
TASK_END
```

This is the dataset update task. It honours the NOASSOC option and takes part in the MULTI_SPEC system.

- If MULTI_SPEC is in use then **TSG_DATASET** is ignored, otherwise it must be given.
- If **UPDATE_ITEMS** is given then it can include one or more of the following space-delimited options. (If not given it defaults to ALL.)
  - **FORMAT** Update the dataset format if necessary.
  - **STSAS** Recalculate System SWIR TSA scalars if necessary.
  - **STSAV** Recalculate System VNIR TSA scalars if necessary.
  - **STSAT** Recalculate System TIR TSA scalars if necessary.
  - **UTSAS** Recalculate User SWIR TSA scalars if necessary.
  - **UTSAV** Recalculate User VNIR TSA scalars if necessary.
  - **UTSAT** Recalculate User TIR TSA scalars if necessary.
  - **CLSS** If necessary, recalculate any CLS scalars that are based on the SWIR TSA reference library.
  - **CLST** If necessary, recalculate any CLS scalars that are based on the TIR TSA reference library.
  - **PREFS** If the TSG setting "On dataset open, check spec calc settings against defaults" in Settings -> System is on, reset the dataset's basic spec-calc settings to the system defaults and recalculate any affected scalars.
  - **ALL** (Default.) Equivalent to FORMAT STSAS STSAV STSAT UTSAS UTSAV UTSAT CLSS CLST PREFS.

For a TSA or CLS item, "if necessary" is when the corresponding TSA version has moved on since the dataset's scalars were last calculated. This generally means that the TSA reference library concerned has changed. This can affect plots and, although it tries, TSG won't resolve all possible issues. For example, if the dataset has any scatter-screen plots that go through a class or set scope based on a TSA class / set then they will probably get knocked around by the change and you ought to revisit them interactively at some point.

Example

```
MULTIOPTIONS noassoc
multifile c:\00me\swift\*.*
task_begin
operation update
update_items format UTSAS
task_end
```

COPYPROC

```
TASK_BEGIN
OPERATION COPYPROC
TSG_DATASET  path+filename of .TSG file (overridden by MULTI_SPEC)
UPDATE_ITEMS  space-delimited options
CPROC_TEMPLATE  path+filename of .TSG template file
CPROC_OPTIONS  space-delimited options
TASK_END
```

This is the copy-processing task. It honours the NOASSOC option and takes part in the MULTI_SPEC system.

- If MULTI_SPEC is in use then **TSG_DATASET** is ignored, otherwise it must be given.
- No, I didn't make a copy-and-paste blunder on this particular occasion. The COPYPROC task spec includes an optional **UPDATE_ITEMS** line and can (first) do dataset updating just like the UPDATE task. See above for a description of UPDATE_ITEMS.
- The **CPROC_TEMPLATE** line must be given. Here you give the path+filename of the template TSG dataset – the one that has the scalars and / or plot layouts that you would like to copy-process. Now did you know – only the template dataset's .TSG file and (if layout copying is to be done) .INI plot-layout files need to be around on the disk? Its .BIP, _CRAS.BIP and _HIRES.DAT are not required.
  Another thing – these days templates can come in associated VSWIR & TIR pairs, like regular datasets. TSG knows how to deal with that automatically. Just keep both .TSG files of the pair together.
- If **CPROC_OPTIONS** is given then it can include one or more of the following space-delimited options. (If not given it defaults to SCALARS LAYOUT.)
  - **TSA** Take the template dataset's TSA settings and, if different, recalculate the current dataset's user-TSA scalars.
  - **SCALARS** Copy-process ***all*** of the template dataset's calculable and compatible scalars.
  - **LAYOUT** Adopt all of the template dataset's layout files. (This is done after the copy-processing of scalars.)
  - **ALL** Equivalent to TSA SCALARS LAYOUT

If MULTI_SPEC is in use then all matched datasets are copy-processed in the same way, using the same template dataset.

Example

```
MULTIOPTIONS swir nodive
multifile c:\00me\swift\0t\*.*
task_begin
operation copyproc
update_items all
cproc_options scalars layout
cproc_template C:\00me\swift\0j\testrocks_tsg.tsg
task_end
```

DOWNSAMP

```
TASK_BEGIN
OPERATION DOWNSAMP
TSG_DATASET  path+filename of .TSG file (overridden by MULTI_SPEC)
DOWNSAMP_CONFIG  path+filename of .TSG template file
OUTPUT_DIR  directory where the downsampling results end up
DSTOLERATE  space-delimited options
TASK_END
```

This is the downsampling task. It *ignores* the NOASSOC option but takes part in the MULTI_SPEC system.

- If MULTI_SPEC is in use then **TSG_DATASET** is ignored, otherwise it must be given. Presently the Downsampling module only deals with one TSG dataset at a time ("primary dataset" philosophy) and has no awareness of primary+assoc dataset pairs. So NOASSOC is ignored and your MULTI_SPEC filtering deserves a fleeting moment's thought.
- A downsampler configuration file must be given in **DOWNSAMP_CONFIG**. You can *obtain* a configuration file by setting up an interactive downsampler session and clicking "Save a downsampler configuration template" in the last page of the wizard. See page 179 of the "what's new" notes.
- If you like, provide a directory for the downsampled files in **OUTPUT_DIR**. Normally they get saved alongside the input datasets. This mechanism lets you collect them elsewhere. It can be handy in a specialised multi-file downsampling run where you'd like all the results to be collected in one place.
- The optional **DSTOLERATE** line has one or more space-delimited options that relax TSG's fussiness when loading the configuration file for one of your datasets. You can tell TSG to tolerate a load problem in any of the first five (now virtual) downsampler wizard pages: The option keywords are:
  - **PAGE1** Selection of the scalar to downsample on, downsampling method, etc. I can't imagine *why* you would want to allow this page to fail, but give the option anyway.
  - **PAGE2** Selection of mask, class & weight scalars, and items to downsample (spectra, scalars, linescan). Think twice about allowing this page to fail. A linescan issue (template wants linescan downsampling but dataset doesn't have it) will actually get tagged to the linescan page.
  - **SPEC** The minimalist page with spectral downsampling options.
  - **SCALAR** Scalar selection and handling. This is the one most likely to "fail". The configuration file includes a complete list of scalars to downsample and it is considered "failure" if *any* of them cannot be found in the current dataset. Perhaps you are willing to go with the subset that's matched?
  - **LINESCAN** Linescan downsampling options. This page will fail if the configuration file calls for linescan downsampling but the current dataset doesn't have linescan.

Example

```
MULTIOPTIONS swir nodive
multifile c:\00me\swift\0t\*.*
task_begin
operation downsamp
downsamp_config C:\00me\swift\downsample.cfg
output_dir c:\00me\0ds
dstolerate scalar
task_end
```

TESTROX

```
TASK_BEGIN
OPERATION TESTROX
TSG_DATASET  path+filename of .TSG file (overridden by MULTI_SPEC)
TESTROCKS_JOB  space-delimited options
BKREM  nothing, GLOBAL or LOCAL
REPORT_FILE  path+filename of report file to generate
TASK_END
```

This task is about reporting on HyLogger wavelength calibration by finding absorption positions of some TestRocks[5] items. It *ignores* the NOASSOC option but takes part in the MULTI_SPEC system. It is only for Vis-SWIR datasets; ideally contemporary TestRocks measurements that include a mylar-on-teflon item.

Update In the past the only datasets handled were HyLogger 2 / 3 TestRocks measurements imported with a 4mm or 8mm pixel size, and HyLogger 1 TestRocks measurements with a 10mm pixel size. In March 2018 the method for locating target items was changed and the task was opened up to **all** Vis-SWIR datasets (not just TestRocks ones). If your dataset includes one or more of the target items (mylar, pyrophyllite, kaolinite, talc) then it can make sense to try the TESTROX task on it.

- If MULTI_SPEC is in use then **TSG_DATASET** is ignored, otherwise it must be given. This task is best run on a collection of TestRocks datasets, however, so MULTI_SPEC is strongly recommended and your MULTIOPTIONS should include these tokens (NOASSOC thrown in for good measure):
  MULTIOPTIONS TESTROCKS VSWIR NOASSOC
- **TESTROCKS_JOB** is mandatory and has no default. One or more of the following space-delimited tokens must be given:
  - **MYLAR** Calculate an average spectrum over the mylar-on-teflon item and report the positions of 7 expected absorptions. An absorption position is found (in the average spectrum) by first looking for a local minimum in the immediate neighbourhood (plus or minus 2 channels) of where the feature is *expected* to be. If a local minimum *is* found then its position is refined by a parabola fit on the 3 channels around the minimum. If an expected feature is not found then its position is reported as 0.
  - **PYRO** Calculate an average spectrum over the pyrophyllite item and report the positions of 6 expected absorptions.
  - **KAOLIN** Calculate an average spectrum over the kaolinite item and report the positions of 3 expected absorptions.
  - **TALC** Calculate an average spectrum over the talc item and report the positions of 11 expected absorptions.
- **BKREM** is optional. It allows you to specify that continuum removal should be done before absorptions are sought. If left out (the default) then the job is done in reflectance space, otherwise you may select one of the following:
  - **GLOBAL** An average spectrum is calculated from the dataset's hull-quotient spectra (instead of reflectance spectra).
  - **LOCAL** An average spectrum is calculated from reflectance spectra, as normal, but a local hull quotient is attempted for each expected feature. (This is an experimental option.)
- **REPORT_FILE** is mandatory. It takes the full filename of the report that you would like the task to put together. Take care – if the file already exists then new entries will be *appended*. The task puts together a CSV table. There's one row for each dataset that was

[5] For those who aren't familiar with the HyLogger world, "Testrocks" is a small panel of known rocks that the HyLogger operator normally measures every day.

processed. The first column has the name of the HyLogger that measured the dataset, the second has the scan date and the third has the filename. After that there's one column for each expected feature, for each TestRocks item specified in `TESTROCKS_JOB`.

Example

```
MULTIOPTIONS swir noassoc testrocks
multifile c:\00me\swift\*.*
task_begin
operation testrox
testrox_job mylar pyro kaolin talc
bkrem local
report_file c:\00me\swift\0rep.csv
task_end
```

WVLCAL

```
TASK_BEGIN
OPERATION WVLCAL
TSG_DATASET  path+filename of .TSG dataset (overridden by MULTI_SPEC)
HWCOPT  nothing, K2206 (default), K2160, Q8625, Q12625, A9200
REPORT_FILE  path+filename of report file to generate
TASK_END
```

This does an empirical wavelength check by finding kaolinite- or quartz-rich samples in the dataset and reporting the position (mean and standard deviation) of a nominated kaolin or quartz absorption. It *ignores* the NOASSOC option but takes part in the MULTI_SPEC system.
Kaolin checking is only for Vis-SWIR datasets, and quartz for TIR. It uses the dataset's TSA results to select suitable samples to check: user TSA results if present, otherwise system TSA. TSA 7.05 results (September 2015 or later) are preferred as they include the "TNorm" TSA scalar, which helps in sample selection. For this reason you might consider running an UPDATE task before WVLCAL.
For kaolin it has four tries at sample selection and gets on with wavelength checking after the first successful try (at least 1 sample accepted): Kaolinite WX (well crystallised) singletons; WX mixtures; PX (poorly crystallised) singletons; PX mixtures. In the mixture tries, samples are not considered if their minor component is from the "white-mica", "smectite" or "other-AlOH" group. In all tries, samples are not considered if their SRSS>=300 or TNorm<0.15 (or just SRSS>=250 if there is no TNorm scalar).
A kaolinite sample's elusive 2206nm (or 2160nm) absorption wavelength is unearthed by feature extraction.
For quartz it has two tries at sample selection: quartz singletons preferably, or quartz mixtures if there are no singletons. Quartz' characteristic 8625nm "absorption" is sought. (Alternatively, you can have it try the lesser 12625nm quartz feature or the 9200nm apatite feature.)

- If MULTI_SPEC is in use then **TSG_DATASET** is ignored, otherwise it must be given. MULTI_SPEC is recommended and your MULTIOPTIONS should include these tokens (NOASSOC thrown in for good measure):
  MULTIOPTIONS VSWIR NOASSOC for kaolin or **MULTIOPTIONS TIR NOASSOC** for quartz
- **HWCOPT** is optional. The task normally reports on the 2206nm kaolin absorption and expects a VSWIR dataset. One may instead use the **K2160** token (viz "HWCOPT K2160") to report on the 2160nm VSWIR kaolin absorption, or the **Q8625** token to report on the **8625nm** quartz feature in a TIR dataset. There are two other experimental TIR options: Q12625 for the lesser 12625nm quartz feature or A9200 for the 9200nm apatite feature. Apatite is rare though.
- **REPORT_FILE** is mandatory. It takes the full filename of the report that you would like the task to put together. Take care – if the file already exists then new entries will be *appended*. The task puts together a CSV table. There's one row for each dataset that was processed. Columns: The name of the HyLogger that measured the dataset, scan date; dataset filename; absorption-wavelength mean; absorption-wavelength standard deviation; number of samples used; sample type (e.g., Pure WX). The first two columns will show "n/a" for non-HyLogging datasets. If none of the sample-selection tries were successful then the dataset's results will be zero and the last column will show "not found".

Examples

```
MULTIOPTIONS swir noassoc
multifile c:\00me\swift\*.*
task_begin
operation wvlcal
report_file c:\00me\swift\0kcrep.csv
```

```
hwcopt k2206
task_end

MULTIOPTIONS tir noassoc
multifile c:\00me\swift\*.*
task_begin
operation wvlcal
report_file c:\00me\swift\0q12crep.csv
hwcopt q12625
task_end
```

PUCKWCAL

```
TASK_BEGIN
OPERATION PUCKWCAL
TSG_DATASET  path+filename of .TSG dataset (overridden by MULTI_SPEC)
REPORT_FILE1  path+filename of report file to generate for primary dataset
REPORT_FILE2  path+filename of report file to generate for associated dataset
TASK_END
```

This task used to be called "TIRPOLYCAL" and only worked on TIR datasets, but it was renamed and upgraded to work on Vis-SWIR and/or TIR datasets.
It *honours* the NOASSOC option and takes part in the MULTI_SPEC system. By all means give it primary+associated dataset pairs, but then be careful to specify two report files in the script.
It does an empirical wavelength check by finding calibration-standard samples in the dataset(s) and reporting the positions (mean and standard deviation) of selected absorptions.

Vis-SWIR

For Vis-SWIR datasets it expects to find spectra of a puck made from spectralon doped with talc and three rare earths, and reports on 25 absorptions.
Here are example results from puck spectra collected from 24 TestRocks measurements.

453_mean	453_stddev	488_mean	488_stddev	524_mean	524_stddev	539_mean	539_stddev
453.327	0.036667	487.784	0.0591036	524.134	0.0446672	539.014	0.0936217

654_mean	654_stddev	744_mean	744_stddev	800_mean	800_stddev	888_mean	888_stddev
653.904	0.0588956	744.871	0.0579322	799.508	0.0345616	887.658	0.0499563

975_mean	975_stddev	1196_mean	1196_stddev	1261_mean	1261_stddev	1322_mean	1322_stddev
975.459	0.23981	1195.96	0.0758734	1260.97	0.0291376	1322.13	0.262735

1392_mean	1392_stddev	1475_mean	1475_stddev	1536_mean	1536_stddev	1643_mean	1643_stddev
1391.97	0.0199514	1474.81	0.0597009	1535.57	0.0191429	1642.53	0.0407821

1683_mean	1683_stddev	1757_mean	1757_stddev	1939_mean	1939_stddev	1971_mean	1971_stddev
1682.91	0.0121566	1756.93	0.0760457	1938.58	0.0387734	1970.8	0.1111

2009_mean	2009_stddev	2289_mean	2289_stddev	2312_mean	2312_stddev	2391_mean	2391_stddev
2008.77	0.257445	2289.02	0.0418918	2311.54	0.0342101	2390.85	0.128416

2468_mean	2468_stddev
2468.46	0.110922

TIR

For TIR datasets it expects to find spectra of a polystyrene-on-gold puck (or something similar like polystyrene ***film*** on aluminium), and reports on 8 polystyrene transmission absorptions. The last one is particularly wide with a poorly-defined minimum so take its result with a pinch of halite.
Here are example results from puck spectra collected from 24 TestRocks measurements.

6245_mean	6245_stddev	6698_mean	6698_stddev	8655_mean	8655_stddev	9347_mean	9347_stddev
6245.3	0.138551	6698.21	0.10576	8655.25	0.134654	9346.8	0.163806

9722_mean	9722_stddev	11018_mean	11018_stddev	11874_mean	11874_stddev	13345_mean	13345_stddev
9721.83	0.17106	11018.4	0.126131	11873.5	0.379664	13342.6	3.52952

Method

First, a correlation match against an internal standard spectrum is used to locate candidate spectra in the dataset. Having found one or more, the wavelengths of the target absorptions are calculated using built-in script scalars. (Most of them do a traditional 3-channel fit to the *reflectance* absorption's lowest 3 points. The scalar scripts can be made available on request.) The wavelength mean and standard deviation are reported for each feature.

- If MULTI_SPEC is in use then **TSG_DATASET** is ignored, otherwise it must be given. MULTI_SPEC is recommended as a calibration report is normally prepared for several datasets. When using MULTI_SPEC with primary+associated dataset pairs, it is important to use **MULTIOPTIONS SWIR** or **MULTIOPTIONS TIR** to avoid handling everything twice and mixing up the reporting.
- **REPORT_FILE1** takes the full filename of the report that you would like the task to put together, *for the primary dataset(s)*. Take care – if the file already exists then new entries will be *appended*. The task puts together a CSV table. There's one row for each dataset that was processed. Columns: The name of the HyLogger that measured the dataset, scan date; dataset filename; pairs of [absorption-wavelength mean, absorption-wavelength standard deviation]; number of samples used. The first two columns will show "n/a" for non-HyLogging datasets. If no suitable spectra were found then the dataset's results will be zero.
- **REPORT_FILE2** takes the full filename of the report that you would like the task to put together, *for the associated dataset(s)*. Do not provide it if you don't have primary+associated dataset pairs, or if you specified the NOASSOC keyword in MULTIOPTIONS.

Example

```
MULTIOPTIONS swir
multifile C:\00me\0cust\Michael\TestRocks\*.*
task_begin
operation puckwcal
report_file1 C:\00me\0cust\Michael\TestRocks\swircalrep.csv
report_file2 C:\00me\0cust\Michael\TestRocks\tircalrep.csv
task_end
```

852LASER

```
TASK_BEGIN
OPERATION 852LASER
TSG_DATASET path+filename of .TSG dataset (overridden by MULTI_SPEC)
TASK_END
```

This is for Vis-SWIR HyLogging datasets only. We use an 852nm laser to keep tight control over the HyLogger2/3's SWIR spectrometer's rotation speed. (The SWIR spectrometer actually "sees" down to this wavelength.) Although we take measures to inhibit it, a small fraction of the laser beam sometimes bounces back off the lenses and beam splitter, and finds its way into the VNIR spectrometer. So it can get *added* to the radiance seen by the VNIR spectrometer. When something about as bright as Teflon (our transfer standard) is measured then the little laser bump is normally ratioed out in the reflectance, but it can have a showing in dark reflectance spectra.
It is not common but this correction can fix it. Spectra are fixed *in place*, and that alone is a caution to run the job only if you need to.
It is for Vis-SWIR datasets only. It ignores the NOASSOC option but takes part in the MULTI_SPEC system. If MULTI_SPEC is used then **TSG_DATASET** is ignored, otherwise it must be given. If MULTI_SPEC is used then MULTIOPTIONS should include these tokens (NOASSOC thrown in for good measure): **MULTIOPTIONS VSWIR NOASSOC**. The **NOFIELD** option is recommended too.

Example

```
MULTIOPTIONS swir nofield
multifile c:\00me\swift\*.*
task_begin
operation 852laser
task_end
```

TPICGEN

```
TASK_BEGIN
OPERATION TPICGEN
TSG_DATASET  path+filename of .TSG dataset (overridden by MULTI_SPEC)
BRANDING  picture title (80 chars), "plain" pictures only (def: HyLogging Systems)
LOGO_FILE  left-logo-icon path (def: built-in CSIRO icon)
LOGO_RIGHT  N or Y, to use "LOGO_FILE" for the right logo too (def: N for CSIRO icon)
MINDEPTH  starting depth (trays before it are skipped) (def: dataset start)
MAXDEPTH  end depth (trays after it are skipped) (def: dataset end)
WIDTHTRIMPC  extra image width trimming in percent (def: 0)
OUT_PIXPERM  output resolution in pixels-per-core-metre (def: 2000)
RESAMP_METH  NN (nearest neighbour), LINEAR, CUBIC, LANCZOS or SUPER (def)
SUBDIR  subdirectory to create (alongside dataset files) for the pictures (def: none)
FILENAME_EX  HOLEID and/or DEPTH ("decorations" for the output filenames) (def: none)
FILE_FORMAT  JPEG, BMP or PNG (def: JPEG)
JPQUAL  quality factor for JPEG output, 1..100 (def: 75)
PIC_TYPE  PLAIN, SCREEN or STICKS (def: PLAIN)
STICK_PER  TRAY, SECTION or INTERVAL (type of image stick, def: TRAY)
STICKSPAN  stick length in metres, interval stick output only (def: 1)
STICK_HORIZ  N or Y, to render image sticks horizontally instead of vertically (def: N)
MASK_SCLR  mask-scalar name, used for an image overlay (def: none)
MASK_COLOUR  colour index for mask overlay (def: 3 and good luck changing it)
MASK_SOLID  N or Y, to draw the mask overlay solid instead of semi-transparent (def: N)
WHITE_BKGND  N or Y, to give the pictures a white background instead of black (def: N)
TASK_END
```

This task generates tray pictures for a HyLogging dataset. It honours the NOASSOC option and takes part in the MULTI_SPEC system. If MULTI_SPEC is used then **TSG_DATASET** is ignored, otherwise it must be given. If MULTI_SPEC is used then MULTIOPTIONS should include the **NOFIELD** token, and the **SWIR** token is *highly recommended* to prevent the thing from doing two passes (one for the Vis-SWIR datasets and another for the TIR ones). Indeed the only reason not to include the **NOASSOC** option as well is if you are using SCREEN for PIC_TYPE and have a mix of Vis-SWIR and TIR scalars plotted in the Tray screen.

So then, I recommend **MULTIOPTIONS swir nofield noassoc** or occasionally **MULTIOPTIONS swir nofield**.

This task has a vast collection of settings. They correspond with the fields that you see in the interactive tray-picture generation dialog and won't get further mention here. (See the vast "What's new in TSG" document for more info.)

Example

```
MULTIOPTIONS swir noassoc nofield
multifile C:\00me\swift\*.*
task_begin
operation tpicgen
subdir 1msticks
out_pixperm 3000
widthtrimpc 7.5
jpqual 80
pic_type sticks
stick_per interval
stickspan 1
stick_horiz y
```

mindepth 100
maxdepth 150
task_end

SPIMPORT

TASK_BEGIN
OPERATION **SPIMPORT**
FORMAT *type of files to import (currently the only option is ASD)*
WATCHDIR *directory where raw spectrum files will "arrive" from time to time*
MOVEDIR *directory where raw spectrum files get moved to after being imported*
MINFILES *minimum number of compatible files in WATCHDIR for an import (1)*
LOOPSEC *seconds to wait before doing another import round (0 meaning no looping)*
NUMLOOPS *number of loops to do (0 for infinite) (only for when LOOPSEC>0)*
OUTFILE *filename (or stub) of TSG dataset to create or append to*
AUTONAME *N or Y, append a timestamp number to OUTFILE to ensure a new dataset on each import*
APPEND *N or Y, existing output dataset: overwrite it, or append spectra to it?*
DATASETCHAIN *N or Y, have the output dataset take the place of "TSG_DATASET" in subsequent tasks?*
WVLMIN *resampling, start output wavelength (source)*
WVLMAX *resampling, end output wavelength (source)*
CHANS *resampling, number of output channels (source)*
WSAMP *resampling, method (L3 if resampling is done, otherwise none)*
CORRFILE *"throughput correction" CSV file, e.g., Spectralon AbsRef*
CPROCFILE *existing TSG dataset from which to copy scalars and layout*
ASD_SMTHWVL0 *short-wavelength-end smoothing, smooth up to this wavelength*
ASD_SMTHWVL1 *long-wavelength-end smoothing, smooth from to this wavelength*
ASD_SMOOTH *degree of smoothing: NONE, LOW, MED, HIGH, EXTRA.*
ASD_SMTHAUTO *N or Y, let TSG decide on the above 3 smoothing parameters?*
ASD_DSTEP *N or Y, do the inter-detector step correction?*
ASD_SUNLIGHT *N or Y, consider interpolating over water-vapour regions of the spectrum?*
ASD_PARSEFN *parse items out of each spectrum's filename and import as scalars. Options: NONE, DEPTH, SAMPLENO*
ASD_FNDEPCH *number of characters assigned to depth when PARSEFN=DEPTH (5)*
ASD_SERIAL *N or Y, import instrument serial number and integration count as scalars*
ASD_GPS *N or Y, import GPS (if present) as scalars*
TASK_END

This task does unsupervised spectral imports *of field-spectrometer data* and it is here to help automate certain workflows. It is a bit like the DOWNLOAD task seen earlier. Like that task, it ignores the NOASSOC option and doesn't take part in the MULTI_SPEC system. It imports spectra to a new dataset or appends them to an existing one using TSG's "dynamic import" system. Currently it only imports field spectra and indeed only ASD binary spectrum files. It is quite a complicated task so its parameters will be discussed in sections.

Basics

The task checks your **watchdir** directory for any files that may be imported. If there are any files there then they are examined for suitability (see **format**). If enough pass examination (see **minfiles**) then the import goes ahead: Spectra are imported to a new TSG dataset or appended to an existing one (see **append**) and, if **movedir** is given, *all* of the files in **watchdir** are moved to **movedir**. If a serious problem was encountered along the way then the task finishes, otherwise if **loopsec** is not zero and **numloops** permits, it sleeps for the given time before trying another import.

- **FORMAT** (opt) The type of spectrum file to import. Currently there is only one option: ASD.

- **WATCHDIR** The directory where spectrum files are expected to arrive. This parameter is mandatory.
    - `Watchdir` may not include any subdirectories, and the output TSG dataset(s) must not be created there.
- **MOVEDIR** (opt) The directory where *all* files in **`watchdir`** are moved to after an import has been attempted.
    - If **`movedir`** is not given then the files are not moved, otherwise they are moved whether or not the import worked.
    - In the case of duplicate filenames an index is appended, as necessary, to prevent files from being overwritten.
    - If there is a file that stubbornly refuses to be moved (e.g., because some program has it open) then the task ends.
- **MINFILES** (opt, default 1) The minimum number of good spectrum files to be found in **`watchdir`** for an import to go ahead (otherwise nothing is done, including file moving).

Looping

This task got its own "looping" support before the **BATCHLOOP** option (*see page 2*) was added to the overall script system. If you only have a SPIMPORT task in your script then you can use either looping method, but if you have a script with two or more tasks (SPIMPORT followed by another task or two) then you will probably want the BATCHLOOP method.
With looping you can have Headless TSG camp out in the background, waking up every so often to check if there is something to do.

- **LOOPSEC** (opt, default 0) The number of seconds for the task to "sleep" before trying another import. A value of 0 turns off looping.
- **NUMLOOPS** (opt, default 0) The number of loops – the number of times to try an import. 0 means infinite.

Output

These parameters are involved: **`OUTFILE`**, **`AUTONAME`** and **`APPEND`**
The task can create a new TSG dataset for imported spectra, or it can append spectra to an existing dataset. Things can get a bit complicated when **looping** is in play. The following workflows are supported for looping:

1. Always import to a new TSG dataset.
    - Provide a partial filename in **`outfile`**. That is, do not include a .TSG extension.
    - Set **`autoname`**=Y. For each import TSG will append `_nnnnnnn.tsg` to the filename and create a new dataset (`nnnnnnn` being derived from a timestamp).
2. Start out with an existing TSG dataset and always append imported spectra to it.
    - Provide a complete TSG dataset filename in **`outfile`**.
    - Set **`append`**=Y
3. Start out with nothing. Let the first import create a new dataset (if it has to), and have subsequent imports append to it.
    - Provide a complete TSG dataset filename in **`outfile`**.
    - Set **`append`**=Y
4. Let each import overwrite the same TSG dataset. (TSG won't stop you…)
    - Provide a complete TSG dataset filename in **`outfile`**.

Note

When appending spectra to an existing dataset, TSG ignores many script settings (resampling, scalar imports etc) and takes this information from the dataset's saved settings instead.

Output chaining

There's this weird option **DATASETCHAIN**. It can be useful in a multi-task script, e.g., one that has a SPIMPORT task followed by a DOWNSAMP task. If you have DATASETCHAIN **Y** then the **TSG_DATASET** entries in all following tasks will be changed to be SPIMPORT's output dataset. That is, all tasks after the SPIMPORT task will be made to work on the dataset that SPIMPORT created / updated. Also, *if SPIMPORT didn't actually import any spectra then all following tasks will be skipped*.

Wavelength resampling

By default the task will take the spectra as they come, preserving their full range & resolution and only resampling if the incoming wavelength increment (from one channel to the next) is not constant. However you can set up resampling if you like. Specify your desired output wavelength range with **wvlmin** & **wvlmax**, and number of output channels with **chans**. Select the resampling method with **wsamp**:

- **LINEAR**=linear interpolation;
- **SPLINE**=spline interpolation;
- **GAUSS**=convolution with "critically sampled" Gaussian bandpasses (Gaussian FWHM = output channel spacing);
- **L3**= Lanczos resampling;
- **DYNL3**=dynamic-window Lanczos resampling (good for going from wavenumbers to wavelengths or vice-versa).

Other options

- You might have seen a thing called "final correction spectrum" in TSG's Import wizard, and perhaps even used it. No? Well it can be handy. If your spectra are reflectance relative to spectralon and you happen to have the spectralon's (*smoothed*) absolute reflectance calibration curve in a CSV file then you can have the import do an approximate correction to absolute reflectance. Provide your CSV filename with **CORRFILE**.
- If you want calculable scalars and layouts to be "copy-processed" from some template TSG dataset that you have, specify this template dataset with **CPROCFILE**.

ASD-specific options

Almost all of the **ASD_** options (shown above) have counterparts in the "ASD" page of the import wizard. See the document tsg8_importing_ASD.pdf.
The smoothing options need some words though. TSG will normally set up smoothing parameters according to the integration time of the first spectrum file that it sees. If you want to set your own smoothing parameters with **ASD_SMTHWVL0**, **ASD_SMTHWVL1** and **ASD_SMOOTH**, you must also set **ASD_SMTHWAUTO=N**.

Examples

This example just has one task, a SPIMPORT task that does its own looping. It creates (if necessary) or appends to a named dataset (0try.tsg).

```
task_begin
operation spimport
loopsec 30
numloops 100
minfiles 10
watchdir C:\00me\asd\0w
movedir C:\00me\asd\0m
outfile C:\00me\asd\0try.tsg
append y
cprocfile c:\00me\swift\repo1.tsg
```

```
corrfile C:\CProjects\0tsg\devbits\those\spectralon.csv
wunits nm
wvlmin 380
wvlmax 2500
chans 531
wsamp L3
format asd
asd_parsefn sampleno
asd_serial y
task_end
```

This example has a **spimport** task followed by a **downsamp** task. If the **spimport** task actually does any importing then it comes up with a unique filename suffix to create a new dataset (each time) thanks to **autoname**. Because of **datasetchain**, the **downsamp** task works on the TSG dataset that gets created by the **spimport** task (its tsg_dataset parameter is effectively ignored), but is automatically skipped if the **spimport** task does nothing. The *whole* script is looped according to batchloop at the start.

```
batchloop 30,20
task_begin
operation spimport
minfiles 10
watchdir C:\00me\asd\0w
movedir C:\00me\asd\0m
outfile C:\00me\asd\0hltry
autoname y
datasetchain y
cprocfile c:\00me\swift\0t\testrocks_1_tsgtray.tsg
corrfile C:\CProjects\0tsg\devbits\SWIR_upd\monica_2018\spectralon.csv
wunits nm
wvlmin 380
wvlmax 2500
chans 531
wsamp L3
asd_smthauto y
asd_parsefn sampleno
task_end
task_begin
operation downsamp
tsg_dataset C:\00me\anything.tsg
downsamp_config C:\00me\swift\downsample.cfg
output_dir c:\00me\0ds
dstolerate scalar
task_end
```

CLIMPORT

```
TASK_BEGIN
OPERATION CLIMPORT
FORMAT  type of files to import (currently HCI3, SDF or SDS)
INPUT_DIR  directory containing a drillhole's worth of files to import
OUTPUT_DIR  directory where the import's TSG dataset should be created
CPROCFILE  existing TSG dataset from which to copy scalars and layout
DATASETCHAIN  N or Y, have the output dataset take the place of "TSG_DATASET" in
subsequent tasks?
HCIXMM  (HCI import) width (across scan) in mm of spectral extraction tile (8)
HCIYMM  (HCI import) length (along scan) in mm of spectral extraction tile (8)
HCIOMM  (HCI import) horizontal shift away from centre-of-scan for extraction tile (0)
JPQUAL  JPEG image compression quality factor (0..100;  75)
HLSCREATE  list of commonly-used core-logging scalars to create on import (system)
SEMSTART  number of samples to mask off at section start (SecEndMask scalar;  system)
SEMEND  number of samples to mask off at section end (SecEndMask scalar;  system)
FMDIL  final-mask-scalar dilation amount (FinalMask scalar;  system)
WVLMIN  resampling, start output wavelength (source)
WVLMAX  resampling, end output wavelength (source)
CHANS  resampling, number of output channels (source)
WSAMP  resampling, method (L3 if resampling is done, otherwise none)
TWVLMIN  TIR resampling, start output wavelength, HyLogger-3 SDS import (system)
TWVLMAX  TIR resampling, end output wavelength, HyLogger-3 SDS import (system)
TCHANS  TIR resampling, number of output channels, HyLogger-3 SDS import (system)
TWSAMP  TIR resampling, method, HyLogger-3 SDS import (system)
SDSCHUNK  spectrum "chunking up" factor for the HyLogger-2/3 SDS import (system)
SDFREPAV  N or Y, repeat-measurement averaging for the (chip mode) HyLogger-1 SDF
import (system)
SDFOPT  image handling and other general options for the HyLogger-1 SDF import (system)
SDSOPT  image handling and other general options for the HyLogger-2/3 SDS import
(system)
SDSCRIT  which errors are critical in the HyLogger-2/3 SDS import (system)
SDSTHRESH  various error thresholds for the HyLogger-2/3 SDS import (system)
SDSTEFALB  reference Teflon albedi for the HyLogger-2/3 SDS import (system)
SDSTHERMC  TIR diag thresholds for the HyLogger-3 SDS import (system)
SDSDNOISE  noise thresholds for the HyLogger-3 SDS import (system)
SDSALBRAT  Vis/SWIR albedo ratio threshold for the HyLogger-2/3 SDS import (system)
SDSSHARP  image sharpening factor for the HyLogger-2/3 SDS import (system)
SDXPANHT  "Vampire Snail Attack" height threshold for a HyLogger SDF/SDS import
(system)
SDXGAMMA  image adjustment gamma for a HyLogger SDF/SDS import (system)
SDXIMGRATE  image output rate (lines per spectrum) for a HyLogger SDF/SDS import
(system)
SDSPDTRAY  N or Y, generate tray datasets (one per tray) instead of a drillhole dataset,
HyLogger-2/3 SDS import
TASK_END
```

This task does unsupervised spectral imports *of core-logging data*. It has a different flavour to the SPIMPORT task above. While SPIMPORT focuses on dynamic imports where a few spectra at a time are imported to one growing dataset, CLIMPORT only creates new datasets. Given that it normally takes quite a long time to import just one core-logging drillhole dataset, CLIMPORT's purpose is to

batch up unsupervised imports of *several* datasets, freeing you up to do other things. (This could even be your chance to finish writing that paper.)

Defaults

The majority of the parameters listed above include the orange text *(system)*, indicating that their defaults come from "the system". Moving on then. No? Well, "the system" means what your TSG remembers from the last time you did an interactive core-logging import, especially a HyLogging import. You can also get at these "system" settings via TSG's File -> Special -> Defaults for SDS and SDF imports menu. So think about it. Do you trust your system settings, or would it be better to write a detailed, self-contained headless script for churning through the import of a few dozen drill-holes? Also note that settings defined in a CLIMPORT headless script will only be "live" for the script – they will not be saved back over the system defaults.

Input and output

This task deals with directories rather than files.

Input

As input the task is given a directory which is expected to contain one drillhole's worth of files. You can either provide this directory explicitly with the **INPUT_DIR** parameter, or you can let the multi-spec system take care of things instead: Here you use MULTIOPTIONS **DIRS** to make the system look for directories rather than files, and you tell it where to start looking with MULTIFILE.
For example I have many drillhole directories in `e:\image_data\corescan\` and I would like to import each drillhole to a TSG dataset of its own. I use this multi-spec (which overrides any INPUT_DIR):

```
MULTIOPTIONS dirs
MULTIFILE e:\image_data\corescan\*.*
```

Caution: By default the multifile system will keep on 'diving' down every subdirectory that it comes across. If you want it to look only one level below your spec then use its nodive option.

Output

The task expects to be run with a multi-spec, normally, and doesn't let you provide a name for the output TSG dataset. It names this dataset according to the input directory. E.g., say the drillhole directory `e:\image_data\corescan\ddh131\` is currently being dealt with. The output dataset will be named `e:\image_data\corescan\ddh131_prof.tsg`.
What you *can* do is (keep the automatic output filename but) have it a different directory, using the **OUTPUT_DIR** parameter, e.g.,

```
OUTPUT_DIR d:\profile_data\cs_import\
```

This output directory must exist. It can be on a different drive to the input.
Another thing you can do (like with the SPIMPORT task) is pass the name of the created TSG dataset on to other tasks in your script by using **DATASETCHAIN Y** For example, say you would like to import a number of datasets (using multi-spec) and you would like tray pictures to be generated for each imported TSG dataset. You would have two tasks on your script – first a CLIMPORT task and then a TPICGEN task. Having **DATASETCHAIN Y** in the CLIMPORT task would pass the name of the newly-created TSG dataset on to the TPICGEN task.

Common parameters

Spectral resampling

If you like you can resample the spectra using some or all of the **WVLMIN**, **WVLMAX**, **CHANS** and **WSAMP** parameters. (See in SPIMPORT above for details.)

- For HyLogger-1 SDF the incoming spec is 350:2500@1 nm and the normal default is resampling to 380:2500@4 nm using the L3 method.
- HyLogger-2/3 SDS the incoming spec is 380:2500@4 nm and the normal default is to take it as-is.
- For Corescan HCI-3 the incoming spec is commonly 448:2500@4 nm and it is all taken by default.

HyLogger TIR

There are separate **TWVLMIN**, **TWVLMAX**, **TCHANS** and **TWSAMP** parameters for resampling HyLogger TIR spectra, should you have them. Incoming spectra are in wavenumbers and the exact coverage varies between instruments. The normal default is to resample to 6000:14500@25 nm using the Dynamic L3 method.

Scalar creation

The import will always create the basic set of core-logging scalars: Tray, Section, SecDist, Depth and so on.
If you like you can have some other common scalars created at import time by using the **HLSCREATE** parameter with one or more options:

- **ProfMin** Creates the "prof_min" scalar – profilometer minimum (used in masking)
- **SecEndMask** Creates the "sec_end_mask" scalar (used for masking). This scalar is off for samples at the start and / or end of a core section, otherwise on. Use the **SEMSTART** parameter to control how many samples get masked off at the start of a section (default 1), and **SEMEND** to control how many at the end (default 1).
- **Kahuna** Creates the "kahuna" junk mask scalar. It requires coverage of the [1304,2496] nm wavelength interval, and should be given ProfMin and SecEndMask.
- **FinalMask** Creates the "final mask" scalar as a copy of kahuna (requires Kahuna). Also, passing a number greater than 0 in **FMDIL** will cause the final-mask scalar to be dilated (more 'off' samples) by that amount.
- **IDL** Initialises the depth-logging system (likes FinalMask).
- **RecRate** Creates the "recovery_rate" scalar (Likes IDL).
- **VirtSec** Creates the "virtual_section" scalar (good for image-style scatterplots).
- **RelRange** Creates the "relative range" spectral-contrast scalar.

If you do not specify HLSCREATE then you will get the **defaults**: ProfMin, SecEndMask, Kahuna, VirtSec
You can also use the **CPROCFILE** parameter to copy layouts and calculable scalars from a template dataset.

HyLogger SDF and SDS parameters

Common

- **SDXIMGRATE** is the desired output image rate in lines per *output* spectrum. The normal default is to take the full incoming resolution. If you use this in an SDS import then take care if also using **SDSCHUNK**.
- **SDXPANHT** is the height above tray bottom, in mm, below which the "Vampire Snail Attack" correction starts taking effect. (The actual correction is enabled separately.)
- **SDXGAMMA** is the image colour stretch gamma power. (The actual correction is enabled separately.)

HyLogger-1 SDF

SDFREPAV

This is a Boolean with value Y or N. e.g., **SDFREPAV Y**
It controls what to do when importing chip-tray data where there's more than one spectrum per chip bucket. N=preserve the multiple spectra; Y=average down to one spectrum per bucket.

SDFOPT

This one includes one or more items (in any order) to enable various SDF import options. The full set looks like this:

SDFOPT DEPEX BLACK WHITE TTCOR WTRIM OLADJ DOZER REVLON PANNY COMNT DOGAM DOTIR TIRCAL DOIMG

You must either leave this parameter out entirely (for defaults), or provide it with ***all*** the options that you'd like enabled. Here's a brief summary of what each option does.

- DEPEX gets the import to create from-depth and to-depth scalars. (Handy for chip trays.)
- BLACK enables the image dark correction.
- WHITE enables the image white correction (important).
- TTCOR enables the image "tan theta" correction, which standardises the pixel size across scan.
- WTRIM enables automatic width trimming of the output linescan raster.
- OLADJ enables the main image-frame-overlap correction.
- DOZER enables a correction that smooths image albedo across frame joins.
- REVLON enables a cross-fade tweak to further conceal frame joins.
- PANNY enables the Vampire Snail Attack correction, which isn't quite as genial as it sounds.
- COMNT gets the import to create a class scalar containing any per-tray comments entered by the HyLogger operator.
- DOGAM enables the image gamma correction.
- DOTIR is the master switch for importing accompanying TIR spectra, should you have them.
- TIRCAL enables TIR temperature correction when importing TIR spectra.
- DOIMG is the master switch for importing imagery.

HyLogger-2/3 SDS

SDSCHUNK

It takes a number. e.g., **SDSCHUNK 2**
It causes the import to "chunk up" (average) incoming spectra. You get fewer output spectra of course but they have better SNR. The normal setting is 2 – chunk up by 2 – yielding an 8mm-per-sample output datastream from a 4mm-per-sample input one.

SDSSHARP

It takes a number 1 to 100. e.g., **SDSSHARP 44**
It's the sharpening amount for when the image sharpening adjustment is enabled (see below).

SDSOPT

It includes one or more items (in any order) to enable various SDS import options. The full set looks like this:

SDSOPT DOIMG BLACK WHITE OLADJ REVLON PANNY WTRIM SHARP TTCOR DOGAM DOLOC DOTIR TIRCAL DSTEP

You must either leave this parameter out entirely (for defaults), or provide it with ***all*** the options that you'd like enabled. Here's a brief summary of what each option does.

- DOIMG is the master switch for importing imagery.
- BLACK enables the image dark correction.

- **WHITE** enables the image white correction (important).
- **OLADJ** enables the main image-frame-overlap correction.
- **REVLON** enables a cross-fade tweak to further conceal frame joins.
- **PANNY** enables the Vampire Snail Attack correction.
- **WTRIM** enables automatic width trimming of the output linescan raster.
- **SHARP** enables an image sharpening adjustment.
- **TTCOR** enables the image "tan theta" correction.
- **DOGAM** enables the image gamma correction.
- **DOLOC** enables an important image-synchronisation step that locates a known edge in the incoming imagery.
- **DOTIR** is the master switch for importing accompanying TIR spectra, should you have them.
- **TIRCAL** enables TIR temperature correction when importing TIR spectra.
- **DSTEP** enables a VNIR-SWIR step correction. It should only be enabled if you know that the data have been measured with the HyLogger's own step correction turned off.

SDSCRIT

This one works like SDSOPT above. It includes one or more items (in any order) to enable "fail on error" for various SDS diags, and you must either leave it out entirely (for defaults) or specify ***all*** of the items you want enabled. The full set looks like this:

SDSCRIT MAXCL DARKMAX WHITEMINA WHITEMINM MAXSAT WHITESTEP WHITECLIP TEFALB ALBRAT TIRGEN TIRSCAT TIRBKTMP TIRTMTMP TIRTDTMP TIRTMP DNOISE

- **MAXCL** fail on too many crystal-lock errors.
- **DARKMAX** fail if the dark image cal isn't dark enough.
- **WHITEMINA** fail if the white image cal isn't white enough (average DN).
- **WHITEMINM** fail if the white image cal isn't white enough (minimum DN).
- **MAXSAT** fail if the imagery over Teflon is too coloured.
- **WHITESTEP** fail if there's too big a local DN difference in the white cal.
- **WHITECLIP** fail if the imagery over the white cal is clipped.
- **TEFALB** fail if the raw instrument response over Teflon is out of bounds.
- **ALBRAT** fail if the VNIR/SWIR albedo ratio is out of bounds.
- **TIRGEN** fail if the TIR temperature-correction stage reports a general error.
- **TIRSCAT** fail if the TIR temperature-correction stage reports too much source scattered into the background.
- **TIRBKTMP** fail if the TIR temperature-correction stage reports too much change in background temperature.
- **TIRTMTMP** fail if the TIR temperature-correction stage reports too much change in tray temperature (median).
- **TIRTDTMP** fail if the TIR temperature-correction stage reports too much change in tray temperature (standard deviation).
- **TIRDTMP** fail if the TIR temperature-correction stage reports too much change in tray temperature (any sample).
- **DNOISE** fail if the (new) noise-on-dark-calibration check finds that a tray's noise measure is too high.

SDSTHRESH

This one accesses *some* of the SDS import diag thresholds. It has one or more name=value pairs, e.g., `MAXSAT=17 DARKMAX=25`. Items can be in any order, and unlike with the fussy `SDSOPT` and `SDSCRIT` above, you can specify just the items that you want to change from their defaults. Any items that you don't specify will be left at their defaults.

- **MAXCL=** gives the maximum number of crystal-lock errors (%) per section.

- **DARKMAX=** gives the brightest allowed dark image cal (max DN).
- **WHITEMINA=** gives the darkest allowed white image cal (average DN).
- **WHITEMINM=** gives the darkest allowed white image cal (minimum DN).
- **MAXSAT=** gives the maximum allowed image colour saturation % over Teflon.
- **WHITESTEP=** gives the maximum allowed step (local DN difference) in the white cal.
- **WHITECLIP=** is really a toggle (value 0 or 1) that snuck in as a "threshold". If enabled (**WHITECLIP=1**) then TSG checks for DN clipping over the white cal.

SDSTEFALB

Use this to specify your HyLogger's typical raw spectral albedo over Teflon. It takes up to three name=value pairs (with V=for VNIR, S= for SWIR and T= for TIR):

```
SDSTEFALB V=15273.8 S=0.03233 T=0.0345
```

Only the ones you specify will be changed – the others will be left at their default values.
Special note: If you give a value of 0 for any of these then the corresponding albedo will be drawn from the first tray of each drillhole that you import. This is probably the best course of action when re-importing historic data, scanned when the HyLogger's albedo thresholds were different to its current ones.

SDSTHERMC

This item accesses diag thresholds to do with TIR correction. It takes up to five name=value pairs, e.g.,

```
SDSTHERMC TIRSCAT=0.02 TIRBKTMP=10 TIRTMTMP=10 TIRTDTMP=15 TIRDTMP=20
```

Only the ones you specify will be changed – the others will be left at their default values.
The thresholds are:

- **TIRSCAT:** max source scattered into the background.
- **TIRBKTMP:** max change in background temperature.
- **TIRTMTMP:** max change in tray temperature (median).
- **TIRTDTMP:** max change in tray temperature (standard deviation).
- **TIRDTMP:** max change in tray temperature (any sample).

SDSDNOISE

This item accesses spectrometer noise thresholds. For each tray, a noise measure is estimated over the tray's dark calibration target. This is done for both the SWIR and TIR spectrometers, although only the SWIR spectrometer has raised concerns to-date. (The default TIR threshold should not trigger.)
An over-threshold noise measure will stop the import with an error if the DNOISE critical bit is set. SDSDNOISE takes up to two name=value pairs, with S= for SWIR and T= for TIR. The example below sets both of the thresholds to TSG's default values:

```
SDSDNOISE S=0.0.00004 T=0.0001
```

SDSALBRAT

Use this to specify the VNIR/SWIR albedo sanity ratio for your HyLogger.
E.g., SDSALBRAT 3.5

SDSPDTRAY

This is a Boolean with value Y or N. e.g., **SDSPDTRAY Y**
When enabled, the importer does not create a TSG dataset for the drillhole, or follow the output dataset naming convention discussed earlier. Instead, it creates a dataset (or dataset pair) for each tray, named after the .SDS file and ending with "_tsgtray". So it essentially recreates the little single-tray TSG datasets that one normally gets from the HyLogger operator.
Note this option is **not compatible** with the **DATASETCHAIN** option.

Corescan HCI-3 parameters

Profile extraction

TSG's Corescan import makes a HyLogger-like *profile* dataset from a Corescan HCI-3 hyperspectral *image* dataset. (A profile dataset is just a single spectrum wide.) You can control the sampling window – how much hyperspectral imagery is averaged down to one profile spectrum:

- **HCIXMM** gives the width in mm of the sampling window (across scan) in mm (default 8)
- **HCIYMM** gives the length in mm of the sampling window (along scan) in mm (default 8)
- **HCIOMM** gives the offset in mm of the sampling window (across scan from the middle of the scan, plus or minus) in mm (default 0 meaning middle of scan)

For example:

```
HCIXMM 16
HCIYMM 10
HCIOMM -8
```

This configuration will give a sampling window that's 16mm wide and 10mm long. Instead of being positioned at the centre of the scan, it'll be positioned left of centre – the right edge of the window will be at the centre of the scan. Given that the normal HCI-3 spectral cell size is 0.5mm, this window will take 16*2*10*2 = 640 HCI-3 spectra and average them down to 1 profile spectrum.
HCI-3 2D profilometer data get sampled through the same window across scan, but preserved along scan. (As in a normal HyLogger dataset there will be many profilometer points per spectrum along scan.)
HCI-3 RGB image data are kept at full resolution, but compressed.

Examples

This script has two tasks. The first imports a HCI-3 drillhole dataset and the second generates image sticks (one per metre) for the imported dataset. It does this for all drillhole directories that it finds in the MULTIFILE spec.

```
MULTIOPTIONS dirs
MULTIFILE E:\corescan\area52\*.*
task_begin
operation climport
format hci3
output_dir C:\00me\swift\csprof
hcixmm 8
hciymm 4
jpqual 80
datasetchain y
task_end
task_begin
operation tpicgen
subdir 1msticks
out_pixperm 3000
widthtrimpc 7.5
jpqual 80
pic_type sticks
stick_per interval
stickspan 1
stick_horiz y
task_end
```

This script does a HyLogger3 SDS import of one specific drillhole. To be independent of system-remembered SDS settings it includes all relevant SDS import options, settings and thresholds except – *notably* – the SDSTEFALB raw-albedo values. It has these set to zero to instruct the import to derive their actual values from the first input tray.

```
task_begin
operation climport
format sds
input_dir c:\00me\swift\0t
output_dir C:\00me\swift\0i
sdschunk 2
sdxgamma 0.72
jpqual 80
sdsopt doimg black white oladj revlon wtrim ttcor dogam doloc dotir tircal
sdscrit maxcl whitestep albrat
sdsthresh maxcl=20 darkmax=17 whitemina=160 whiteminm=125 maxsat=17 whitestep=40
whiteclip=1
sdstefalb v=0 s=0 t=0
sdsthermc tirscat=0.02 tirbktemp=10 tirtmtmp=10 tirtdtmp=15 tirdtmp=18
sdsalbrat 3.5
task_end
```

SHELLEXEC

```
TASK_BEGIN
OPERATION SHELLEXEC
VERB  operation / action for the ShellExecute command (default is OPEN)
FILE  file or object on which to execute the given VERB (mandatory)
PARAMS  If FILE specifies an executable file, PARAMS is a string that specifies the
parameters to be passed to FILE (optional, default none)
DIRECTORY  default working directory (optional, default none meaning current directory)
SHOW  N or Y, allow the application concerned to show its window.  (Why you want to have
this turned on in headless mode is beyond me, but do it if you must.)
WAIT  N or Y, have Headless TSG wait for the application concerned to finish / close
TASK_END
```

This task spawns another process from headless TSG using the ShellExecute command. It was put into the headless system mainly to allow you to pick up where TSG left off, e.g., to handle (with some other program that you have) a CSV file produced by the DOWNSAMP task.

- The most commonly-used **verb** is open (the default). Allowed verbs will depend on file, and will generally be shown in file's right-click menu.
- **File** will normally be the path+filename of an executable to run, but can instead be a document that Windows knows how to open. E.g., Windows knows how to open .html, .txt and .csv files (amongst others).
- If file is an executable then you can use **params** to pass space-separated arguments to it. If file is a document then do not specify **params**.
- Normally, headless TSG will move on after launching the SHELLEXEC task, allowing the launched process to carry on in parallel. If you would prefer headless TSG to wait for the process to finish before moving on, specify **wait y**.

Dataset filename substitution

There may be times when you'd like to use the SHELLEXEC task on some TSG dataset file that a previous task has worked on, but you don't have the dataset's filename handy. The cases currently handled are multi-spec scripts and the SPIMPORT task (which might have autoname y). In these cases you can use the special keyword **$TSGFILE** for **file** and / or in **params**. TSG will then substitute the current dataset's actual filename for this keyword.
If you use it in params then it is good practice to put it in quotes, i.e., "$TSGFILE". (If you don't, TSG will attempt to insert quotes itself if the actual filename includes spaces.)
e.g., (for some imaginary program) params –archive "$TSGFILE"

Example

This (dumb) example runs a PUCKWCAL task and opens the resulting report file in the notepad++ text editor. Notepad++ is opened in "topmost" mode. TSG waits until Notepad++ has been closed.

```
task_begin
operation puckwcal
tsg_dataset C:\00me\TestRocks\swirstandard.tsg
report_file1 C:\00me\0look.csv
task_end
task_begin
operation shellexec
verb open
file C:\Program Files\Notepad++\notepad++.exe
params –alwaysOnTop C:\00me\0look.csv
show y
wait y
```

```
task_end
```

COPYSCLR

```
TASK_BEGIN
OPERATION COPYSCLR
TSG_DATASET  path+filename of primary dataset file (overridden by MULTI_SPEC)
SOURCE  primary or assoc, which dataset in the pair currently hosts the scalar
NAME  the name of the scalar to copy (mandatory)
TYPE  The named scalar's type.  One of:  ANY  MASK  IMPORT  PROF  FEATX  ARITH  CLASSX
SMOOTH  BATCH  CORE  AUXM  PFIT  STAT  PLSP
```

This task is only for primary+assoc dataset pairs. It copies one scalar from the primary to the associated dataset, or vice-versa. It works for numeric, class, mask and RGB-colour scalars. The scalar to be copied is identified by name, and this name matching can optionally be reinforced by type matching. If the scalar already exists in the target dataset then it is overwritten, otherwise it is created. Whatever its type in the source dataset, the copied scalar ends up as an IMPORT (not recalculable) type in the target dataset.

COPYSCLR takes part in the MULTI_SPEC system but *ignores* the NOASSOC option. When using MULTI_SPEC, it is necessary to qualify it with **MULTIOPTIONS swir** or **MULTIOPTIONS tir**. It is crucial that you are clear on which dataset of the pair currently has the scalar to be copied across. If COPYSCLR is given a dataset that does not have a paired counterpart then it does nothing.

Why this task? TSG has some restrictions on what can be done with scalars in a paired-dataset environment. For example the expression and batch scalar methods can only work with scalars from the same dataset, and the downsampler deals exclusively with one dataset. So (for example) if you create a mask in the VSWIR dataset of a VSWIR-TIR pair, you can use the COPYSCLR method to copy the mask to the TIR dataset and then use this mask in TIR downsampling.

Why optional type matching? Identifying a scalar by name is usually good enough but sometimes maybe not. Say you create class-extraction scalars called 'plag' in your TIR datasets and would like to use the scalars in VSWIR work. Now you're thinking that although 'plag' is indeed an excellent name for this class-extraction scalar, perhaps you have used it for other scalars sometimes. To prevent some accidents you can tell headless TSG to look for a *class-extraction* scalar named 'plag'. Taking this to a more interesting level, say you've done a round of work and copied your 'plag' scalar, and now you have refined your TIR analysis, updated 'plag' in the TIR datasets, and want to copy the updates over to the VSWIR datasets. For whatever reason you write a new script and make a mistake with the MULTIOPTIONS line, specifying swir instead of tir. Type matching will save you. TSG won't obliterate the TIR 'plag' class-extraction scalars with the stale old copies from the VSWIR datasets because the ones in the VSWIR datasets have import type.

Example

This example copies a mask scalar called SWIR_ASPECTRAL from VSWIR datasets to their TIR counterparts.

```
MULTIOPTIONS swir
multifile c:\00me\swift\*.*
task_begin
operation copysclr
source primary
name swir_aspectral
type mask
task_end
```