Scientific Computer System

User's Guide

Version 5.1.0

2023

Office of Marine and Aviation Operations
Marine and Aviation Cyber Center
Software Engineering Division

1315 East-West Highway, 10th Floor, Silver Spring, MD 20910

Website - SCSshore.noaa.gov

Overview of the Scientific Computer System (SCS)

General Information

The Scientific Computer System (SCS) is a data acquisition and display system designed for Oceanographic, Atmospheric, and Fisheries research applications. It acquires sensor data from shipboard oceanographic, atmospheric, and fisheries sensors and provides this information to scientists in real time via text and graphic displays, while simultaneously logging the data to disk for later analysis. SCS also performs quality checks by monitoring I/O, providing delta/range checks and plotting data after acquisition.

System Overview

The Scientific Computer System (SCS) was designed and implemented based on a user requirements study completed in 1986. The first version of SCS was certified in 1989 on the NOAA Ship MALCOLM BALDRIGE. The general requirements are summarized below:

- Provide real-time data acquisition for an extensive suite of sensors.
- Provide real-time displays and graphics to the scientist.
- Provide real-time quality checks on the data being collected.
- Provide logging of the real time data to disk for data post-processing at sea and in the laboratory environment.

SCS provides this functionality through a set of programs that work in concert to acquire, log and display the data. The system is structured in such a way that each program performs a single function and may be run on independent servers. Most client interactions are conducted via a modern web browser. This simplifies the maintenance of the system while maximizing its flexibility, long-term use, platform independence and

reliability by isolating user interface displays from mission critical data collection functions. The programs interconnect using a client/server architecture based on SignalR, Windows Communication Foundation (WCF) and various web technologies which allow the display interfaces to run remotely client computers with the real-time data being provided via SignalR from the acquisition computer.

System Requirements

Recommended Setup

Servers

- Two (2) identical servers (SCS-A and SCS-B) with one (SCS-A) being the primary application server and the other (SCS-B) being the primary database server.
- Each server should have at least 32GB of RAM
- Each server should have enough disk space to accommodate your data - RAID5 generally acceptable, Storage Spaces Direct recommended if large.
- Each server should be running Windows Server 2016 or higher with all patches applied.
- Server websites should be secured with HTPS/HSTS and valid (or trusted) certificates.

> If you have spare funds the best place to put them would be into additional RAM. The more memory you add the better performance you will get. Generally speaking disk speed/space and CPU on a generic server are sufficient for the demands of SCS however being able to utilize RAM for caching will yield the most benefit.

> SCS is currently written using the .NET 4.8 Framework and requires a host capable of running it (Windows). Future versions of SCS will target the .NET Core framework instead, allowing for the server to be Windows, Linux or a variety of other operating systems, but as of the time of this writing the server must be Windows based.

> ⚠ Creating a NLB cluster to distribute the load and allow for scalability sounds ideal, however this is not our default setup in the NOAA fleet and has not been tested. With the future of the fleet moving towards virtualization and potentially even on-prem PaaS we will not be investigating this as an option.

Clients

Clients connect over the web, so any client with an up to date and modern web browser should work with SCS. SCS was designed to target clients with resolutions of around 1600x1200 but the UI is responsive and will adjust to smaller resolutions. While in theory it will work on mobile devices it has not been tested as of yet so UI difficulties may arise if accessed in this manner.

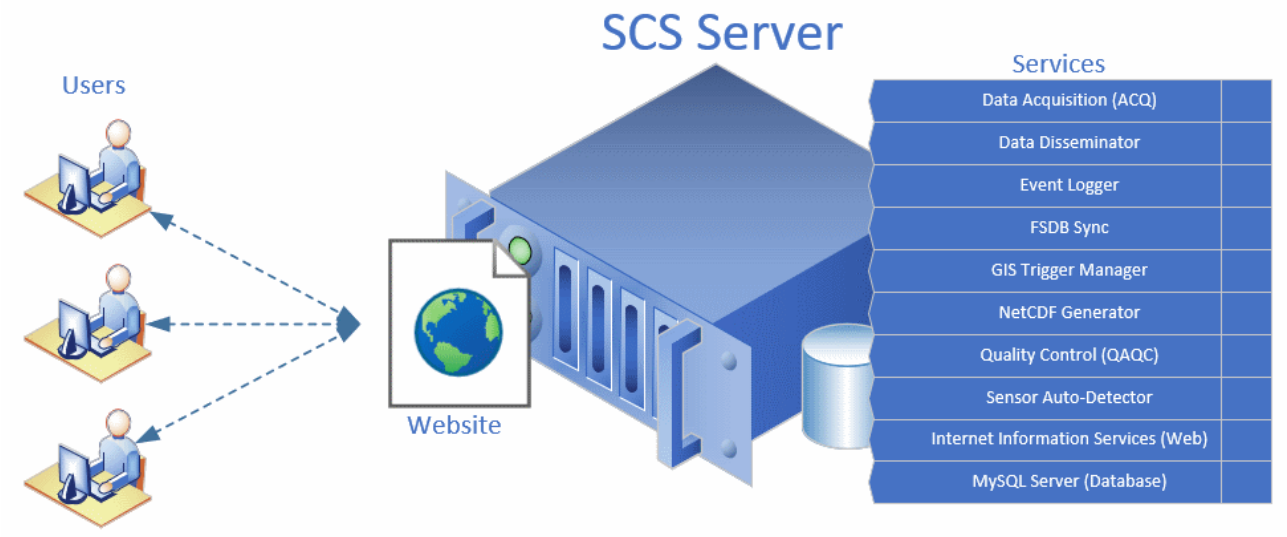
Minimum Setup

While the more resources you can allocate to the servers the smoother SCS will run, an expensive box is not a requirement. SCS can run inside a virtual machine hosted on a normal developer's laptop with minimal RAM and CPU. However, if you have a large load with a large number of sensors or high data rates you will

run into latency and timeout issues if your machine cannot accommodate. If you have a nominal setup with only a few sensors and a few clients then a basic machine running Windows 10 will suffice. As with the recommended setup, adding more RAM to your machine will yield the best impact on performance.

Architecture

SCS has been designed to be as client agnostic as possible both in terms of the client location (network wise) and client operating system. The ideal way to do this was to make SCS a web based application. General users who interact with SCS will do so over a modern web browser, this includes anonymous users, scientists and administrators alike. Of course the core of the system remains application based and is hosted on a centralized server. However, these applications have morphed into services instead and can run independent of any user's profile. This was done to alleviate the issues arising from HSPD-12 and other security measures forcing the use of MFA and requiring a user to be logged in to run SCS. The current SCS can run in the background whether anyone is actually logged into the server or not, and will automatically re-institute itself if the server is rebooted for whatever reason (such as when it's being patched).



All sensor data logged by SCS is saved into the backend database. MySQL was chosen as it performs decently well, has most the features needed and is freely available to all potential users of the software.

IIS is used as the web server to provide the website to remote clients and to host the SignalR hubs used to provide data feeds and inter-process/client communication. IIS comes as a standard part of Windows Server and an express version can be installed on Windows 10 if needed.

Folder Structure

SCS is primarily using MySQL as it's data store. However it does write certain things to the filesystem, by default all these will be written into D:\SCS though that can be changed at installation time.

Of particular interest are:

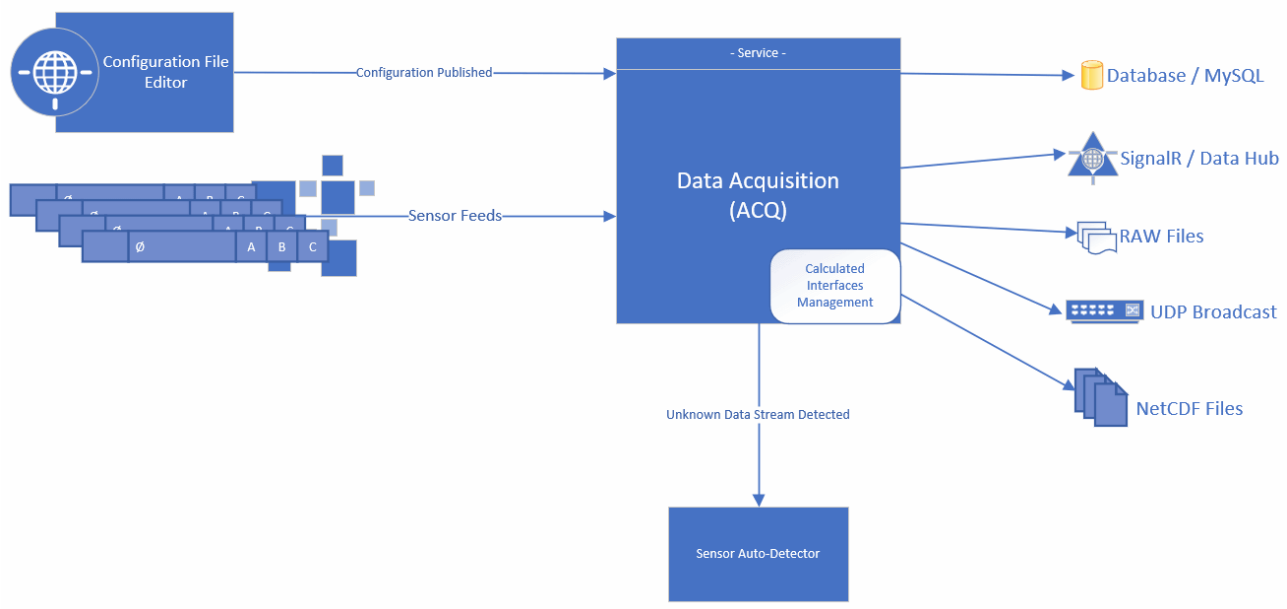
D:\SCS\ConfigurationState	System Folder containing information needed by CFE to maintain the working configuration and track changes.
D:\SCS\Data	Sensor data folder which houses NCEI dumps, the RAW files and the NetCDF exports extracted from the database
D:\SCS\Email	Location of any emails which are queued up waiting to be sent
D:\SCS\Logs	Primary [non-database] location of system logs
D:\SCS\MySqlData	Primary location of database files
D:\SCS\Services	Location of all SCS background services
D:\SCS\Backup	Backups of the database (excludes sensor data!) primarily involving templates
D:\SCS\Temp	Temporary SCS files

Data Acquisition Service (ACQ)

The Data Acquisition Service (ACQ) is the heart of the entire SCS system. It is in charge of collection and storage of all sensor data streams being fed into SCS. This is the service which opens all the COM ports and reads the incoming sensor data before saving it to disk and supplying it for consumption by the other services and client displays.

ACQ gets its sensor configuration from CFE and continuously listens to the Configuration hub for announcements regarding configuration changes. In the event the sensor setup changes ACQ will automatically adjust itself to handle the modifications without having to be restarted. In other words, if you add a new sensor or change something in CFE and publish then your changes should be reflected almost immediately by ACQ without any further action on your part.

As ACQ reads and processes the various data streams defined by CFE it actively pushes the sensor data out to multiple targets. The two primary outputs are the backend database for storage and the SignalR Data hub for client visualizations.



⚠ The ACQ service should always be running if you want to collect data!

In addition to ingesting all the actual real time sensor data ACQ also calculates any derived interfaces you have defined in CFE along with all the SAMOS averages. It actively monitors resource consumption on the host and will disable sensors which start to overwhelm predefined thresholds. If it finds data streams that it does not recognize it will automatically feed those on to CFE allowing the user to decide if they should be included in the master configuration or ignored.

Data Dissemination Service

The Data Dissemination Service is a background service which plays a key role in getting data from the ship out to remote clients. Its job is to handle the larger and less time sensitive type of data traffic.

⚠ This service does not send real-time data out for client visualization, that is the role of the SignalR Data hub.

Email

When the SCS system attempts to send an email it routes all requests through this service. Email has become a very important means of communication, however with the removal of on board mail servers it has been difficult to reliably utilize it as internet connectivity is intermittent. The Data Dissemination Service attempts to resolve this by adding some resiliency to the send process.

Instead of a send-and-forget type approach, the service will attempt to send the email as requested but monitor for failure. In the event of failure the email is encrypted and serialized to storage. At regular intervals the service will attempt to reconnect to the [shore based] mail server, once a connection is successful an attempt is made to retry sending all prior failures. This cycle continues until every emailed queued up is successfully pushed to the mail server.

SCS only deals with SMTP on the ship, no IMAP services are provided.

Email settings can be modified in the Settings portion of the website.

SAMOS

While ACQ is in charge of calculating the data averages for all SAMOS interfaces, the Data Dissemination Service is in charge of taking those outputs and getting them to FSU. If enabled, the service has a nightly job which pulls all data from the prior day and packages it up in compliance with the SAMOS v1 packaging guidelines along with all relevant meta-data, file hashes and needed information. It then pushes the package to shore over to FSU for processing.

In the event a user requests a manual send of data the request is routed and processed through this service.

See also: SAMOS

TSG

While ACQ is in charge of recording the data from all TSG sensors, the Data Dissemination Service is in charge of taking those outputs and getting them to AOML. If enabled, the service has a nightly job which pulls all data from the prior day and packages it up in compliance with the AOML packaging guidelines along with all relevant meta-data and needed information. It then pushes the package to shore over to AOML for processing.

In the event a user requests a manual send of data the request is routed and processed through this service.

See also: TSG

NCEI

While ACQ is in charge of recording the data from all sensors, the Data Dissemination Service is in charge of taking those outputs and getting them to NCEI. If enabled, the service has a nightly job which pulls all data from the prior day and packages it up in compliance with the NCEI NetCDF packaging guidelines along with all relevant meta-data and needed information. It then pushes the package to shore over to NCEI for processing. If applicable SCS may also update the OMAO Ship Daily Activity Tracking system (SDAT) to record the fact that the ship had submitted the day to shore officially transferring responsibility of the submission over to NCEI.

In the event a user requests a manual send of data the request is routed and processed through this service.

> NOAA Ships may switch to using the local netman server and rSync instead of FTP to get datasets off the ship and to NCEI. If that is the case on your vessel then SCS will move the data to the netman server instead of FTP'ing it directly to NCEI. Once it is on the netman server (on the ship) SCS is no longer responsible for getting it to shore as control of that server is outside the scope / control of SCS.

See also: NCEI

Custom Messages

The Data Dissemination Service is in charge of management of all Custom Message templates defined in the system. The service keeps track of all of these templates and enables/disables them as appropriate. It collects the appropriate sensor data, formats it as requested by the user and pushes it out as defined in the template. For this reason the Data Dissemination Service should be whitelisted in the local systems firewall as many of these templates push out via TCP/IP or UDP/IP.

See also: Custom Messages

Event Logger Service

The Event Logger Service is a background service whose job is sole management of all Event related templates and data inside SCS. This service is the one which keeps track of and executes event templates previously defined by the user base. When you start a new event this service loads it into memory and manages all the interactions with all the various people participating. It executes all the action sequences and pulls all the sensor data, essentially it does everything associated with the event at run-time.

> When an Event is started the template as it is specified AT THAT MOMENT is saved along with the Event. If you change the template later those changes are only applicable to instances started subsequently and do not have any impact of previously or currently running events.

The Event Logger Service is robust in such that if it is shutdown gracefully (eg the computer is rebooted) it will save the state of all running events and attempt to bring them back online once the system boots back up.

See also: Events Overview
See also: Event Templates
See also: Event Management

Fleet Sensor Database Synchronization Service (FSDB)

The FSDB Service is an optional background service which attempts to keep your local ship's SCS configuration in sync with a shore based server. This is useful for many reasons.

- Automatically keeps your backup server (SCS-B) in sync with your primary server (SCS-A) so in the event of a failure you can bring your backup online with minimal downtime or effort on your part.
- Creates a centralized shore based collection point providing visibility into the capabilities and sensor suites located on each vessel in the fleet.
- Creates a COOP backup off the ship for your SCS settings.
- Provides a means for 3rd parties such as NCEI, SAMOS, etc to gather your metadata without you having to submit it over the VSAT repeatedly saving your bandwidth.
- Allows shore-based support to pull and use your SCS settings to assist you with any issues you may be facing by replicating your situation quickly on our development servers.

> This feature is only available to the NOAA fleet. If you wish to setup a similar situation for your non-NOAA ship you will have to setup your own shore-based server. If so please reach out to us, the software is free like the rest of SCS and we can assist you with getting it running as time allows.

If you are not one of the large NOAA vessels you will most likely have this service disabled.

> ⚠ Only configuration items are sync's between ship and shore. SENSOR DATA IS NOT TRANSMITTED due to bandwidth limitations. Do not rely on this as a method of backing up your sensor data!

Geographic Information System (GIS) Trigger Service

The GIS Trigger Service is a background service whose job is to monitor the ship's current position and launch predefined actions when certain geographical boundaries are crossed.

This service listens to the system's reference Latitude and Longitude data streams in order to determine the ships position. Disabling ACQ or not specifying a Lat/Lon source in CFE will effectively render this service useless.

Actions are not launched immediately when a boundary is crossed. Instead the service waits until the vessel is 0.25 nautical miles past the boundary before firing the actions. This is to ensure that the ship is indeed past the point and will not traverse back and forth (eg moving slowly in heavy seas or is riding the line, etc) causing the action to be launched repeatedly.

> This buffer distance is adjustable, you can change the GISTrigger.BoundaryCrossingMargin variable in the services config file if you feel this number is to high or to low.

See also: GIS Triggers

NetCDF Generation Service

The NetCDF Generation Service is a background service whose job is to take sensor data and dump it into a NCEI compliant NetCDF file.

This service has three primary ways of generating NetCDF files:

1. Real-time data feed over WCF from ACQ
2. Nightly NCEI job which dumps all data collected from prior day
3. Manual extraction tasks created and managed via the website

> Methods 2 & 3 above pull data directly from the database and are not real-time.

If you disable the NetCDF service sensor data will still be recorded into the database (you will not lose data). However, until you get it running again submissions to NCEI will fail and at the end of the cruise you will not be able to pull your data from the database out into NetCDF files for distribution to the science parties. In this event you could fallback to using the RAW files.

See also: NetCDF

Quality Assurance / Quality Control (QAQC) Service

The QA/QC Service is a background service whose job is to monitor the data recorded by ACQ for anomalies in order to flag bad data and attempt to correct issues as close to the source and as timely as possible.

This service replaces the concept of DataMon in prior versions of SCS. It skims data from the database and runs all QA/QC checks defined which target the messages under review.

> QA/QC results are not currently integrated with the datasets submitted to NCEI or others and are for ship-based review only.

> ⚠ Cleaning up RAW data from the database will delete any QA/QC data related to the removed records.

See also: Monitoring QA/QC

See also: Defining QA/QC

Scheduler Service

The Scheduler Service is a background service which performs scheduled system tasks internal to SCS. Some examples are performing database backups, syncronizing files between the SCS servers for CoOP, maintenace on db indexes and tables, and so forth.

Sensor Automatic Detection Service

The Sensor Auto-Detect Service is a background service whose job is to monitor all COM ports not being used by ACQ for new data feeds which may be of interest to the user. It constantly opens each port and iterates through the various setting combinations of baud, parity, flow control, etc in the hopes of finding inputs which look like normal ASCII text.

In the event it discovers valid input it will attempt to determine what type of input it found, which sensor category/type it would belong to and present it via CFE to the user with option of adding it to the active sensor configuration in SCS. Under ideal circumstances you could plug a new sensor into your Digi/Comtrol and in after a bit of time all the data streams coming from that device would be automatically discovered, configured and awaiting your approval from addition to your sensor suite in CFE. This obviously is a nice-to-have feature, it does not prevent you from manually creating any items you want and can be completely ignored or even disabled without impacting the core functionalities of SCS.

> If you have a minimally resourced host this may be an optional service you would want to disable to conserve RAM and CPU.

See also: CFE - Auto Sensor Detection

Help

If you are reading this then you have already gained access to the first source of help for all things SCS.

This manual is available in multiple formats:

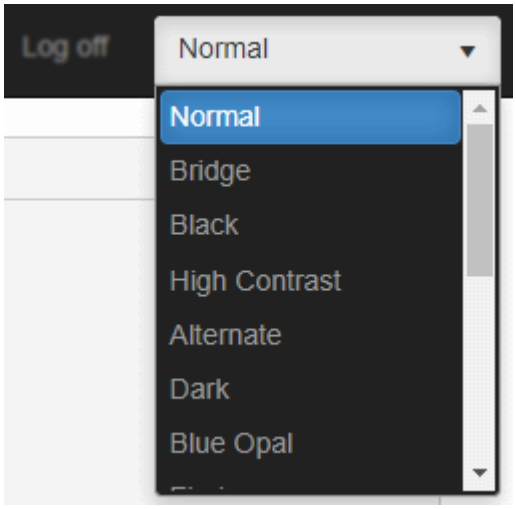
- Online, included in the SCS installation and accessible from your ships SCS local website
- Online, accessible from the SCSShore website
- PDF
- EPub
- Microsoft HTML Help

To obtain a copy of any of the above documentations please visit the SCSShore website which has download links for all the above.

OMAO also conducts multiple SCS training sessions annually. If you would like to attend please reach out to us and we will get you on the list. Be aware, since this is a free (paid for by NOAA) event NOAA employees may consume all available seats. However, if spots are open any and all are welcome to attend. If you are unable to attend one of our courses for whatever reason we can always work with you to send a trainer to you for advice, help with installation and/or for a customized class for whatever duration and covering whatever topics you desire.

Theming of the SCS User Interface

The main SCS interface is presented via a responsive HTML5 website. This website has multiple themes which users can apply to change the look and feel of all the various pages.



There are a variety of preset themes available, to switch from one to another simply click the drown down list located in the upper right corner of the website (next to the Log In / Log Off link) and choose the one that interests you.

While changing themes is mostly a matter of aesthetics there are situations where certain themes are actually beneficial. Perhaps a user would prefer a high contrast UI for 508 compliance, or perhaps users operating on the bridge at night would prefer to filter out bright colors. In these use cases it could be advantageous for a user to select a certain theme and/or avoid others.

However, in the end, choice of theme has no actual effect on the functionality of SCS.

⚠ If you set explicit colors when building your widgets they will override any selected theme! It's advised you never do this unless for a specific reason.

Prerequisites

Hardware

The hardware required is dependent upon the expected load your specific instance will be expected to handle. SCS can run inside a virtual machine hosted on a laptop, but the fewer resources you provide the poorer the performance will be. If you only have a minimal number of sensors and very few clients then you can attempt to use a low-end setup. The more sensors you add, the rate at which they output their data, the number of client computers you will be servicing and/or the amount of calculated feeds or number of QA/QC checks all will dictate the amount of resources required your server. It is generally recommended to have substantially more than you will need as hardware is [relatively] cheap and performance issues can lead to timestamp errors, etc since Windows is not a RTOS.

For general reference NOAA ships base hardware and setup (as of FY20) are noted below:

Platform	(2x) Dell PowerEdge R440 Server
Processor	(2x) Intel® Xeon® Gold 5218 2.3G, 16C/ 32T, 10.4GT/s, 22M Cache, Turbo, HT (125W) DDR4-2666
Memory	(2x) 32GB RDIMM, 2666MT/s, Dual Rank
Disk	PERC H730P RAID Controller (4x) 2TB 7.2K RPM NLSAS 12Gbps 512n 3.5in Hot-Plug Hard Drive

⚠ If getting a 1U server (such as the R440) be sure to have enough PCI slots to accommodate a serial port adapter card if applicable. The spec above includes a 2x16 LP PCI Riser.

While both servers have the full SCS suite installed in normal operation the two servers are assigned different roles. One server is setup as the primary Web/Application server which runs the website and majority of SCS services (data acquisition, event logging, etc). The other server is setup as the primary database server and is essentially dedicated to that job (majority of RAM is assigned to the database).

As mentioned, the full suite is installed on both servers though they are tailored to their specific role. This is because in the event of a hardware failure either server can assume "full" SCS capabilities (eg the web server also has the database and the database server has the web so either can provide SCS CoOp if the other server goes down). However, while SCS can continue to function in the event one of the servers goes down, performance will be degraded until it is restored. Please contact the SEG team immediately if you experience a hardware failure while underway and need assistance getting fully operational again. SOPs and helper scripts have been written to switch the role of any given server to "single" mode should it's partner fail.

Operating System

Currently, Windows remains the required underlying host OS. SCS can be installed on:

- Windows 10
- Windows Server 2016

- Windows Server 2019

> ⚠ All above operating systems should have all updates and service packs applied to bring them up to the latest version prior to installation of SCS.

Microsoft .NET Framework

Minimum version: 4.8

Getting Installation Binaries

SCS has a semi-automated installer which can be used to install all or subsets of the SCS software suite. Before you can proceed with installation you will need a copy of these files. They can be obtained via the 'downloads' portion of the SCS shore based website (https://scsshore.noaa.gov/Home/Downloads). If you have any questions please feel free to reach out to the SCS team (https://scsshore.noaa.gov/Home/Contact) and we will work with you to get the software installed.

> Note, v5 of SCS no longer requires a signed memorandum of understanding (MoU) to download or use the software. Instead you will be required to accept the licensing agreement upon install.

Installing SCS

While installing SCS isn't always trivial, many of the complexities SCS have been minimized through the installation wizard SEG has provided. Installing SCS can be as simple as clicking "Next" repeatedly or moderately complicated if you choose to install on multiple servers.

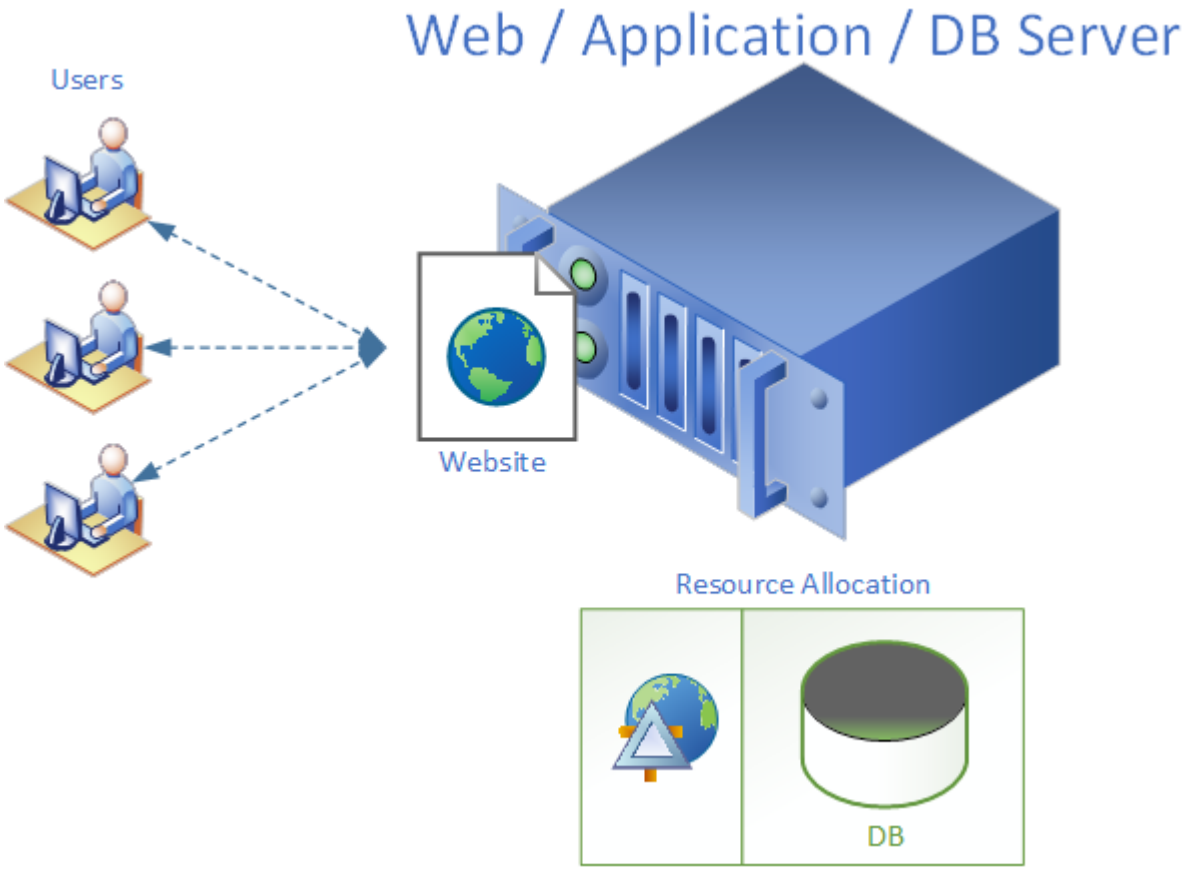
If you have an existing SCS installation which you are replacing it is highly recommended you back it all up prior to proceeding with these instructions. It is also advisable to:

- Perform a ZipShip export (explicit backup of your current sensor configuration)
  - If you're running SCS v4
    - Backup your *Template* directory
    - Backup your *scs.cfg* file
    - Backup your *Datalog40* directory
  - If you're running SCS v5
    - Backup your database
    - Perform an FSDB sync operation in CFE (if applicable)

Single Server Installation

If you want to have everything running on a single server, whether two large servers where one acts as a backup, or if you only have a single box (smaller deployments with lower resource requirements) the installation process is straight-forward.

The single server will have all SCS services, host the website and host the database. The installer will setup the database to have slightly more resources allocated to it than to the web server or SCS services but both will share everything available on the one host. This is a good setup if you only have a single server, or if you want to setup two independent boxes (one in production/active, one as a spare) and is similar to the model used in previous SCSv4 installations (SCS-A and SCS-B).



1. Procure hardware and/or validate that the intended host meets the prerequisites for your use case.
2. Obtain a copy of the latest SCS installation media.
3. Prepare the host and run *setup.exe*
    1. It is recommended you utilized a RAID, a SAN/NAS or implement a software defined version such as Storage Spaces Direct.
    2. It is recommended you create a partition or volume mounted to the "D:\" drive for storage of all SCS related files and data
    3. Open the installation media and launch the *setup.exe* file (you will need Administrative rights on the server to do this)

4. Read and accept the licensing agreement to continue (signed MOUs are no longer required for non-NOAA users)
5. *Select Ship*: Choose your ship from the dropdown list. If you do not see your ship then select *Other Ship* and type her name in the input box which appears below.
6. *Server Setup Type*: For a Single Server Installation be sure to choose *Single Machine*.
7. *Select Drive for Installation*: Choose the drive you setup in Step (1) above as the target for SCS files and logged data
8. *Enable FSDB Sync*: Unless you selected *Other Ship* in Step (6) this should be checked as it enables real-time replication of your configuration with a shore based system for backup and restoration purposes. If you are not a large NOAA ship then this should be unchecked as you generally will not have the shore-side half of the system setup and running to sync with.
9. *Enabled NCEI Submission*: If you wish to use SCS to submit your data to the NOAA archives this should be checked.
10. *Enabled SAMOS Submission*: If you wish to use SCS to submit subsets of your data to SAMOS/FSU this should remain checked (all large-NOAA vessels should leave this enabled)
11. *Enable TSG Submission*: If you wish to use SCS to submit your TSG data to AOML this should remain checked.
12. *SMTP Host*: If you have a ship based mail server you may enter it's DNS name here. NOAA vessels should use the shore-side gmail server smtp.gmail.com. SCSv5 is tolerant of network instability and is able to queue and send any emails which fail to make it to shore when your network re-establishes itself.
13. *SMTP Port*: The port to use when sending email, gmail / TLS requires 587.
14. *SMTP Credentials*: The username and password to use when sending emails via the host specified in Step (13). If you are a NOAA vessel and using gmail be sure to add the domain to your username (eg john.katebini@NOAA.gov) and to use an application password instead of your 'real' password to account for 2-step verification restrictions.

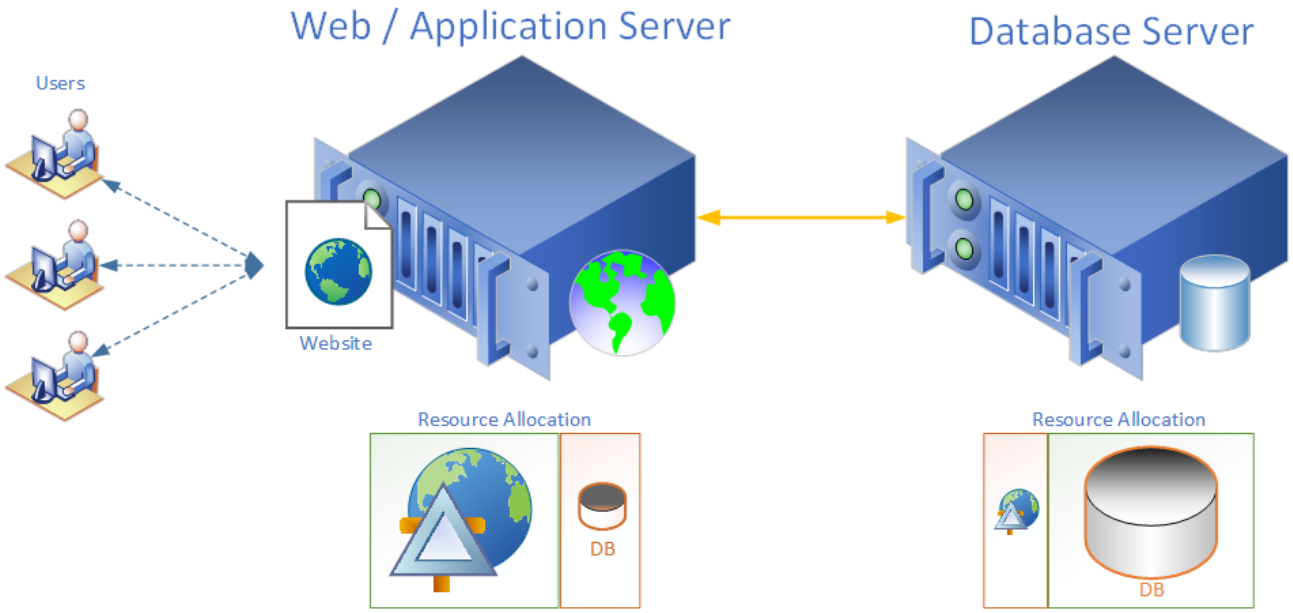
15. The installation wizard will take care of the rest. This can take some time, get some coffee and come back in a bit.

16. Once the installation completes it's recommended you install and use a valid certificate or downgrade your system to HTTP (note HTTPS is required for all US Gov installs) and perform an FSDB sync from CFE (assuming you have a shore side repository of your previous configuration).

Split Responsibility / Multi-Server Installation

If you have two servers the recommended approach is to split the functionality of SCS up between the two of them. This way both servers are utilized while still covering each other in case of a hardware failure. Similar to setting up two Single Server Installations as detailed above, except rather than having the second server as a spare it is actively involved in the production system, thus effectively doubling the compute power you have at your disposal. To accomplish this the installer allows you to setup a machine with a primary focus on being the database or a primary focus on being the website/application server. Both servers will have both roles, however resources will be allocated differently depending on which primary role you have assigned (eg the database server will allocate the majority of memory to the database, etc).

The primary interface for users would be the web/app server which would (behind the scenes) use the database server as it's backend. This offloads all the data storage and querying to the second server freeing up resources on the web/app server to perform better for the users.



Uninstall

The uninstall process for SCS is essentially the same as you would do for any other application installed in windows.

1. Backup all data you might want to keep or image the machine
2. Open Add Remove Programs
3. Find the SCS program
4. Right click and choose Uninstall

Certificates on the SCS Website

A website security certificate is a validation and encryption tool, part of the HTTPS protocol, which secures and encrypts data going back and forth between the server and the client browser. It is issued by a trusted certification authority (CA) who verifies the identity of the owner of a website. The certificate then ensures

the user that the website it is connected to is legitimate and that the connection is safe and secure. - Techopedia

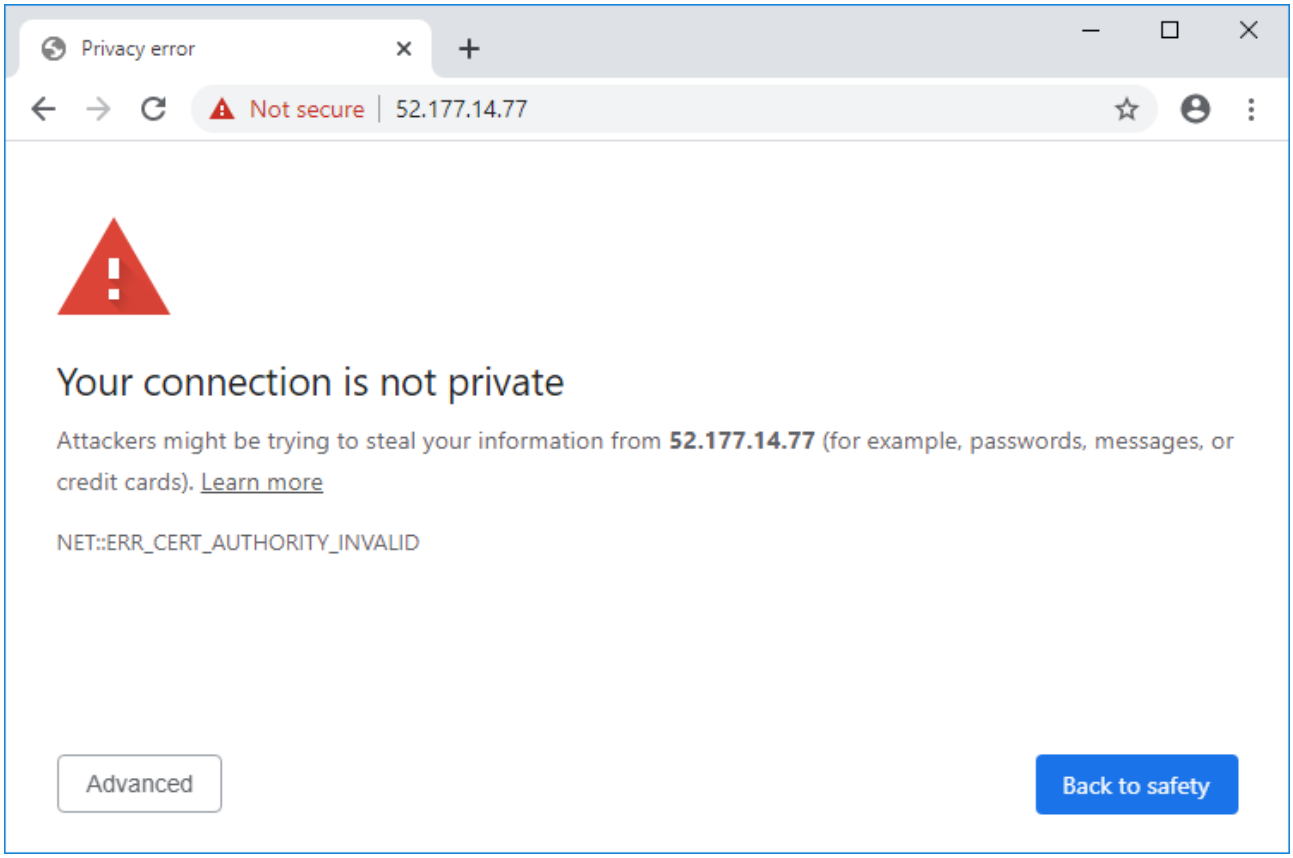
When you browse to a website you use one of two protocols: HTTP or HTTPS

HTTP stands for HyperText Transfer Protocol and allows for communications between a client and a server via request-response. It is the backbone of the world wide web and has been the standard root protocol since the late '90s.

HTTPS stands for HyperText Transfer Protocol Secure and essentially wraps security (TLS or SSL) around normal unsecured HTTP. Traffic moving between the client and the website is theoretically encrypted and secure so sensitive things like usernames/passwords, your bank balances and other such info is protected from 3rd parties trying to listen in. As an evil-doer knowing your bank balance may not seem like a big deal, but if I learn your username/password then your bank balance is now my bank balance, obviously not ideal (for you!).

Because of this, most public websites sites these days utilize HTTPS and require a valid certificate be installed on the source web server and associated with the site. US Government sites are required to use HTTPS as per White House Office of Management and Budget memorandum M-15-13, **"A Policy to Require Secure Connections across Federal Websites and Web Services"** (https://https.cio.gov/)

Many internal sites use an unofficial self-signed certificate. When you browse to the site you get the familiar "insecure site" warning:



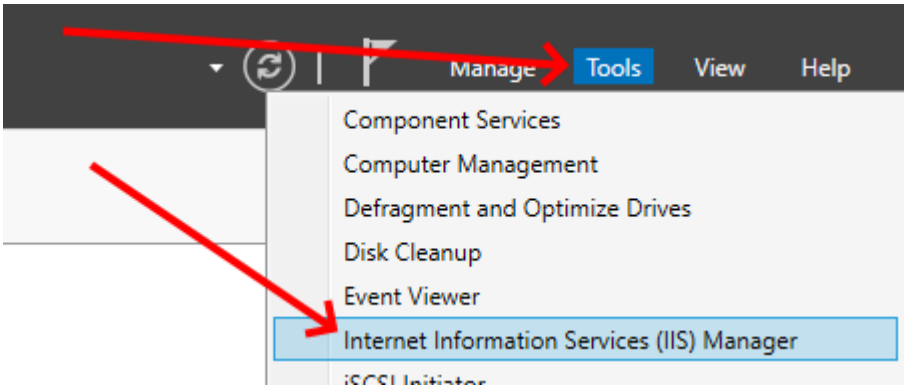
Self-signed certificates are commonly used internally for a variety of reasons. Namely they are free, easily generated, are much more secure then having no certificate at all and don't require a certificate authority (most of which require a website to be internet facing / public). While this is fine for development purposes

when the system is moved into production using a self-signed certificate is a bad idea. Aside from annoying your users it also opens security holes and avenues for man in the middle attacks.

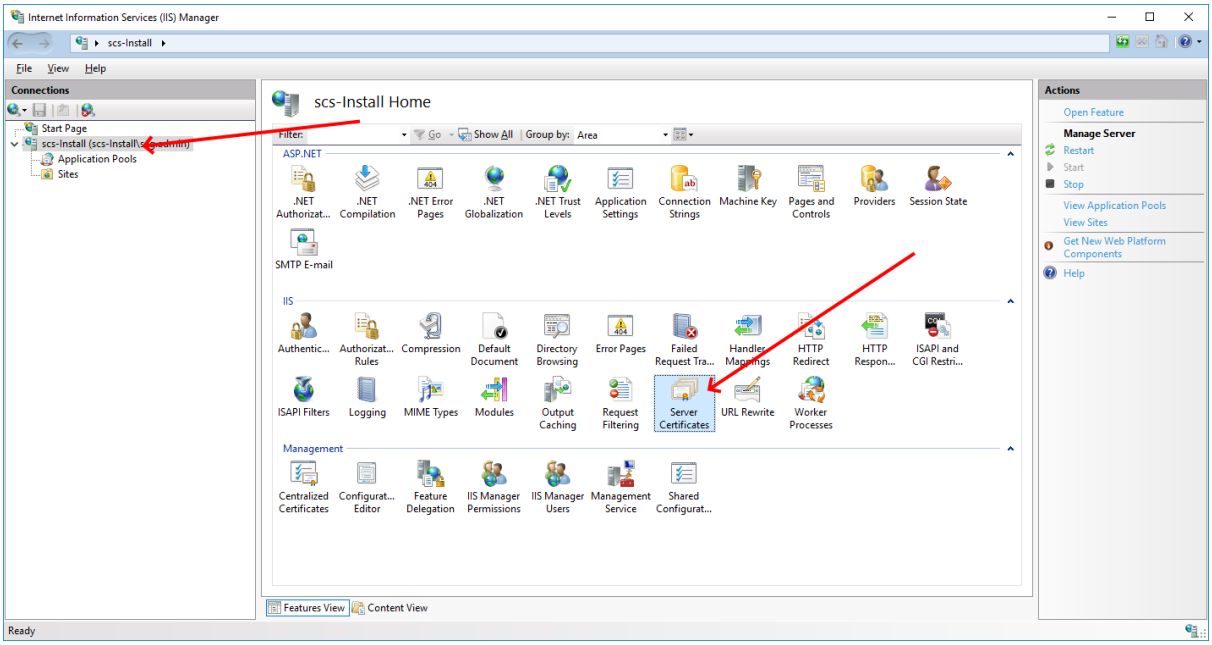
SCS will install by default with both HTTP and HTTPS bindings. While you can access the site using either protocol if you try to use HTTP then SCS will automatically redirect you to the HTTPS version for security purposes.

To associate a certificate with your SCS website

1. Launch Internet Information Services (IIS) Manager. Depending on your operating system this can be done various ways, most windows servers will have it as an option under the Tools portion of Server Manager

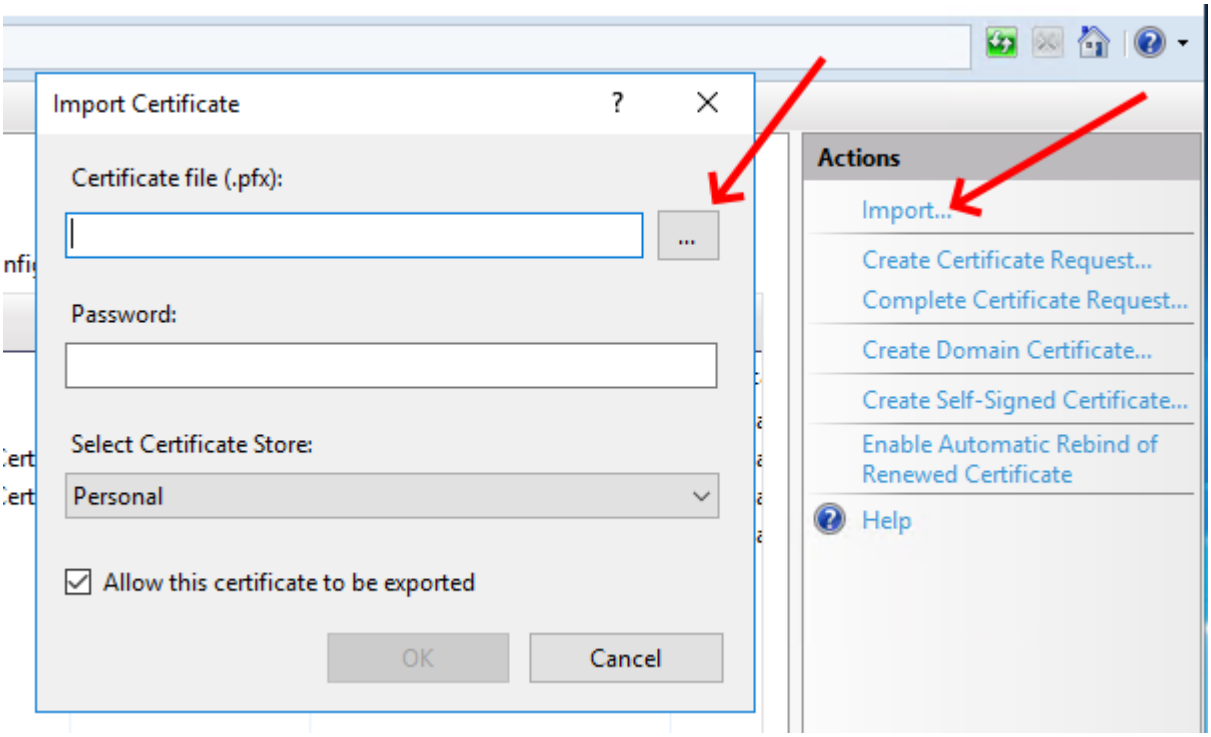


2. Click on the root web role element and double click on the Server Certificates link inside the IIS group.

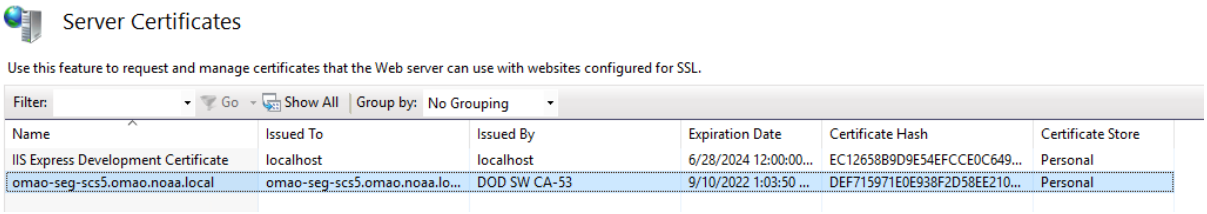


3. In the right pane click the Import action which will open a dialog allowing you to browse to the certificate file and upload it into IIS. Note the password is the password used to secure the certificate when in .pfx format (similar to adding a password to a zip file). If you do not

know the password talk to the source who exported or provided the pfx file.

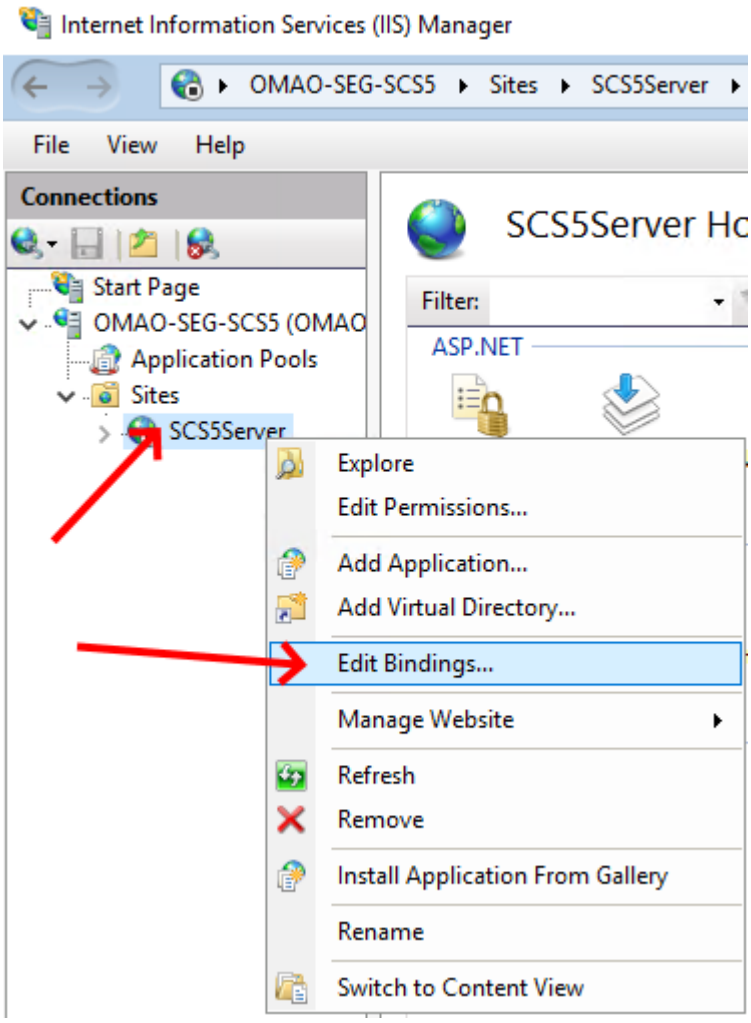


4. Validate that the certificate shows up in the list on the server.



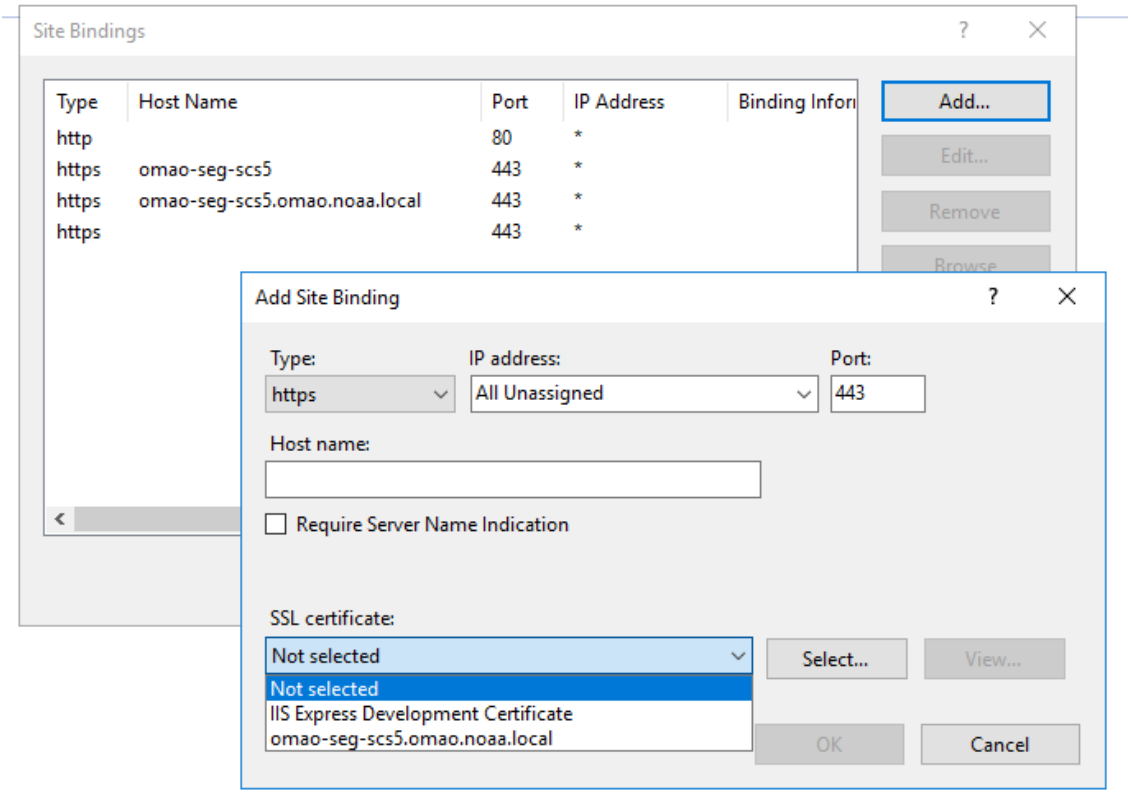
5. Now that the certificate is installed on the server you must associate it with your website itself
    1. In IIS manager, select the SCS website (expand the sites node, the names might be different but it should be the only site on the server)

2. Right click the site and click Edit Bindings



3. Associate the HTTPS binding with the new certificate by choosing it in the dropdown. If you have aliases or other names for the host be sure to enter additional HTTPS

bindings to reflect them.



NOAA Ships

NOAA GFE should use the Department of Defense as their certificate authority (CA). NOAA machines should already have the full DoD certificate chain installed as it's a requirement for Common Access Card (CAC) authentication / login. Each SCS website will have it's own dedicated certificate issued by the DoD. If you do not have a copy of your certificate please reach out to SEG and we will provide one to you.

Non-NOAA Ships

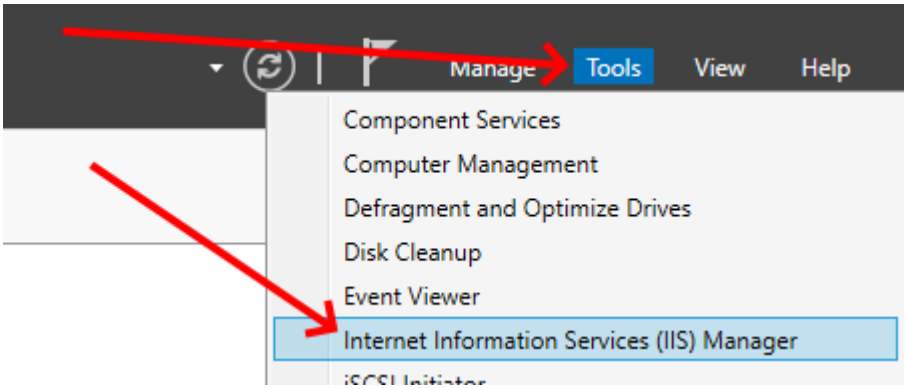
Ships outside the DoD boundary have a few options available to them.

Setup an HTTP / HTTPS hybrid situation

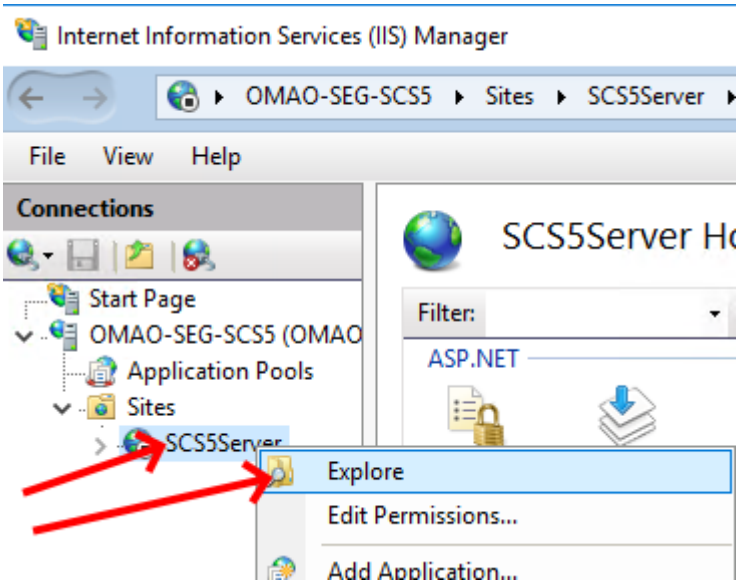
SCS is installed to support both HTTP and HTTPS but is setup in a way which forces the user into HTTPS regardless of which option they choose. If you are unable to obtain a valid certificate and must use a self-signed one then removing that restriction from SCS might be the best way to move forward. The idea would be to use HTTP for normal read-only operation of SCS and use the HTTPS option when doing things that require authentication. This would prevent usernames and passwords being sent in clear text but minimize the number of annoying 'unsecure' warnings you receive from your browser. For the primary workstations you can always explicitly trust the self-signed certificate itself which would take another step towards reducing the red.

> ⚠ It is recommended to use HTTPS whenever possible and definitely when conducting operations that require authentication / login!

1. Switch all SCS services to use the HTTP protocol instead of the HTTPS
2. Launch Internet Information Services (IIS) Manager. Depending on your operating system this can be done various ways, most windows servers will have it as an option under the Tools portion of Server Manager

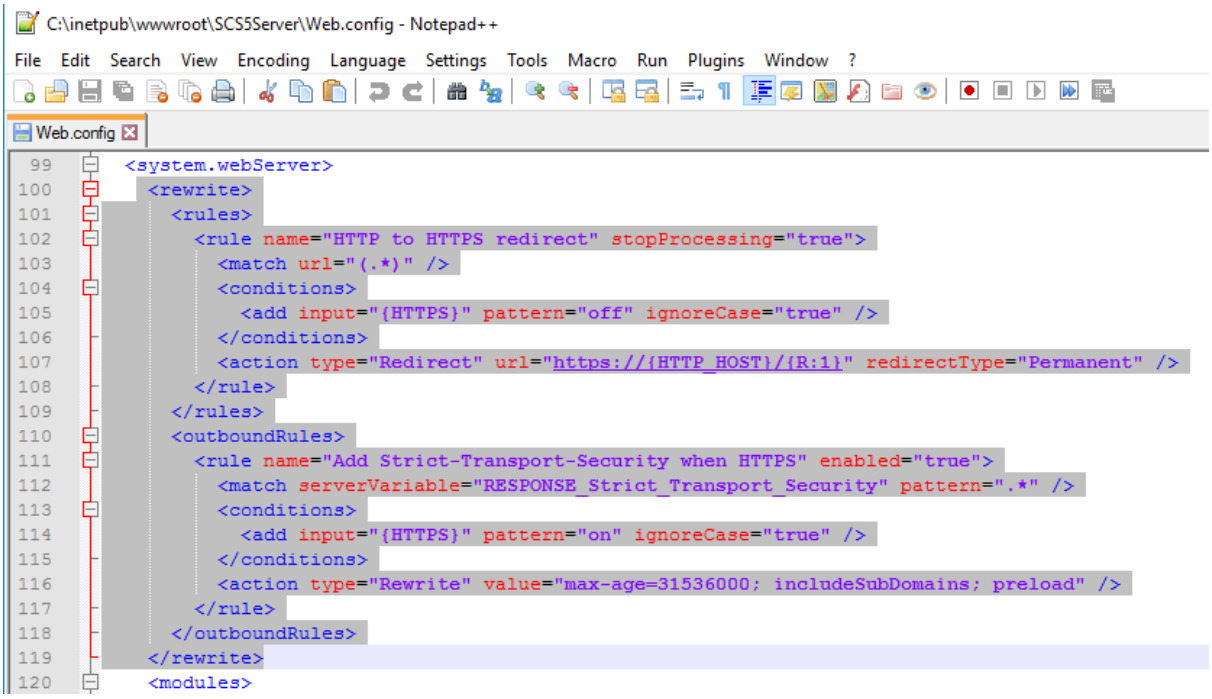


3. Find and expand the SCS website (expand the sites node, the names might be different but it should be the only site on the server)
4. Right click the site and click Explore



5. This will open up windows explorer in the directory containing all the code used to run the SCS Website. Inside this directory is a file named *Web.Config*. Open it using a text editor such as Notepad (you may need to open it in Administrator Mode to save changes). Be careful when editing this file, it might be worth copying it somewhere else (backup) in case things get messed up so you can revert.

6. Search the file for the <system.webServer> node. Inside of it should be a <rewrite> node



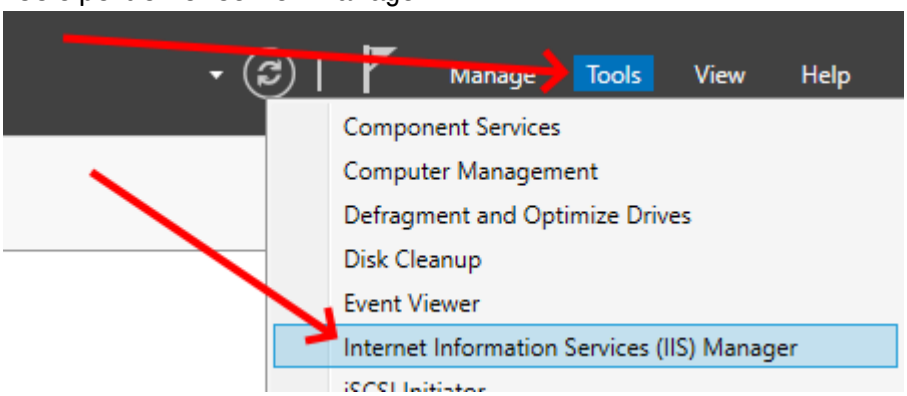
7. Delete the entire <rewrite> node (everything between <rewrite> .... </rewrite> including the <rewrite> tags themselves as highlighted above)
8. Save the *web.config* file.
9. Restart the website

Do not use HTTPS

While not recommended (usernames and passwords will be sent in clear text whenever users work on the site) using standard HTTP is an option. If you have a small setup, completely trust your user base, are disconnected from the internet, etc it is the simplest option available to you.

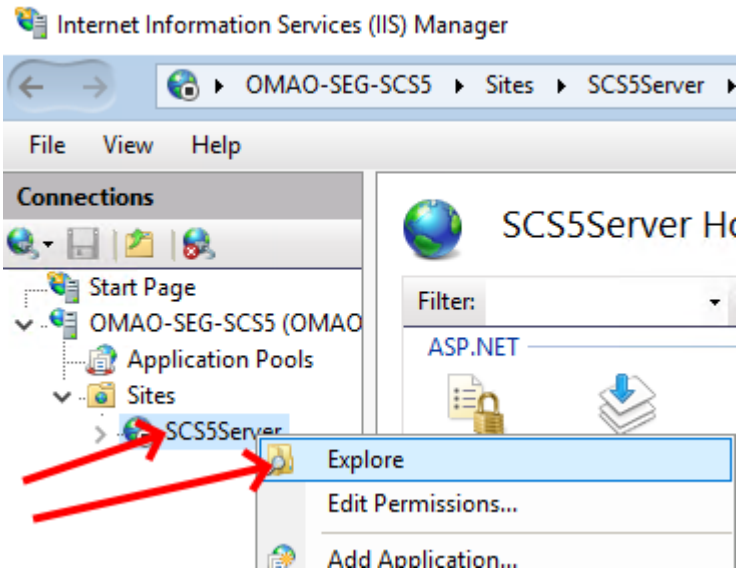
To remove HTTPS from your setup:

1. Switch all SCS services to use the HTTP protocol instead of the HTTPS
2. Launch Internet Information Services (IIS) Manager. Depending on your operating system this can be done various ways, most windows servers will have it as an option under the Tools portion of Server Manager

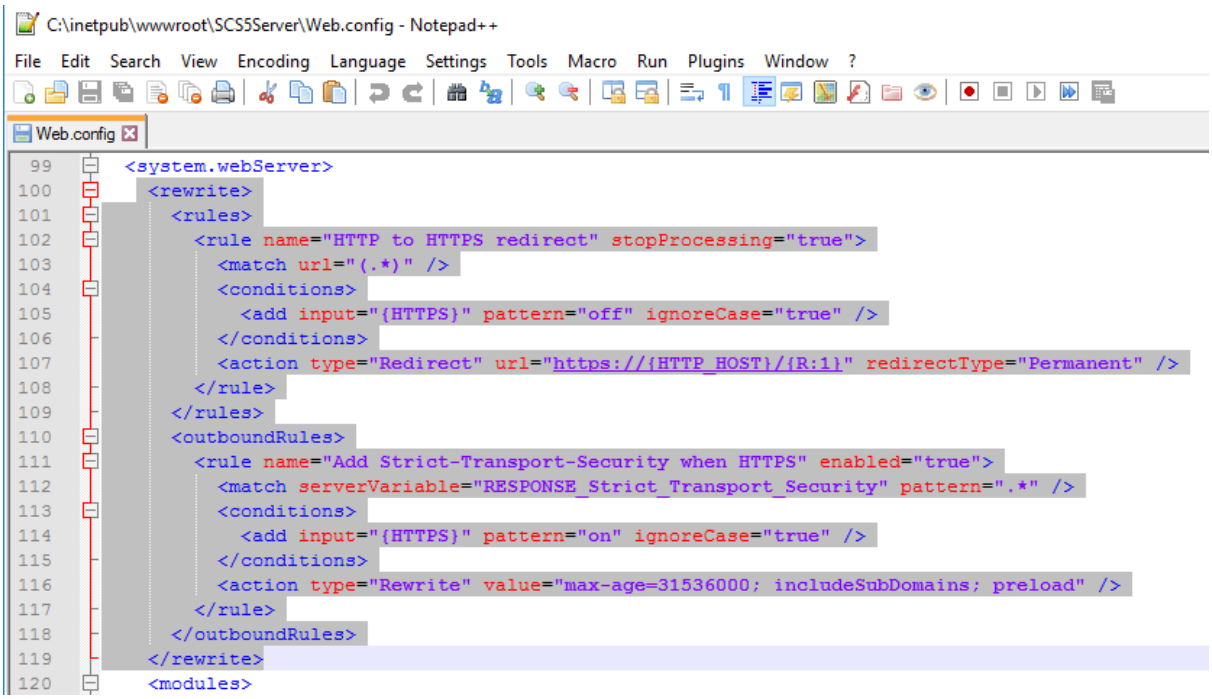


3. Find and expand the SCS website (expand the sites node, the names might be different but it should be the only site on the server)

4. Right click the site and click Explore

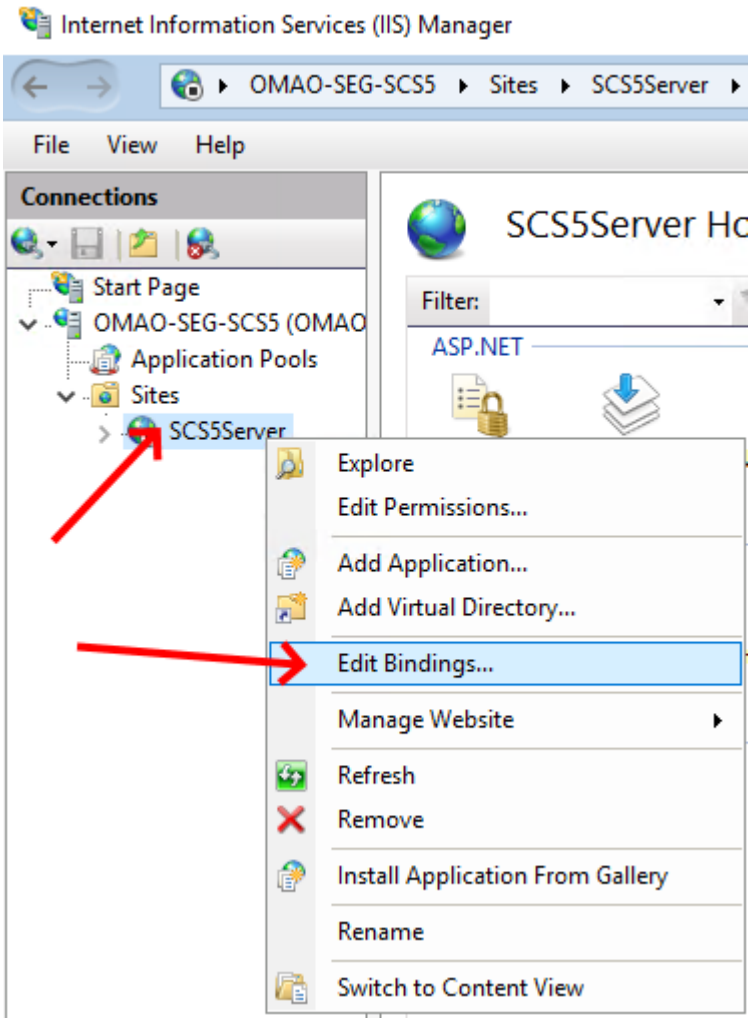


5. This will open up windows explorer in the directory containing all the code used to run the SCS Website. Inside this directory is a file named *Web.Config*. Open it using a text editor such as Notepad (you may need to open it in Administrator Mode to save changes). Be careful when editing this file, it might be worth copying it somewhere else (backup) in case things get messed up so you can revert.
6. Search the file for the <system.webServer> node. Inside of it should be a <rewrite> node



7. Delete the entire <rewrite> node (everything between <rewrite> .... </rewrite> including the <rewrite> tags themselves as highlighted above)
8. Save the *web.config* file.

9. Right click the site and click Edit Bindings



10. You will see a list of bindings associated with the site, remove all HTTPS bindings and hit OK
11. Restart the website

After this there will be no more HTTPS endpoint, users will have to browse to http://[server name or IP here] to access SCS.

Procure a publicly valid certificate

You can always purchase a certificate from a known vendor such as VeriSign, RapidSSL, DigiCert, etc. This is the solution most every website on the internet takes. However, as a ship you are quite unlike most websites. Be aware these certificates are meant for internet facing sites, so unless your exposing your server to the internet and assigning it a valid publicly resolvable DNS name this is unlikely to be a feasible option.

Setup your own CA

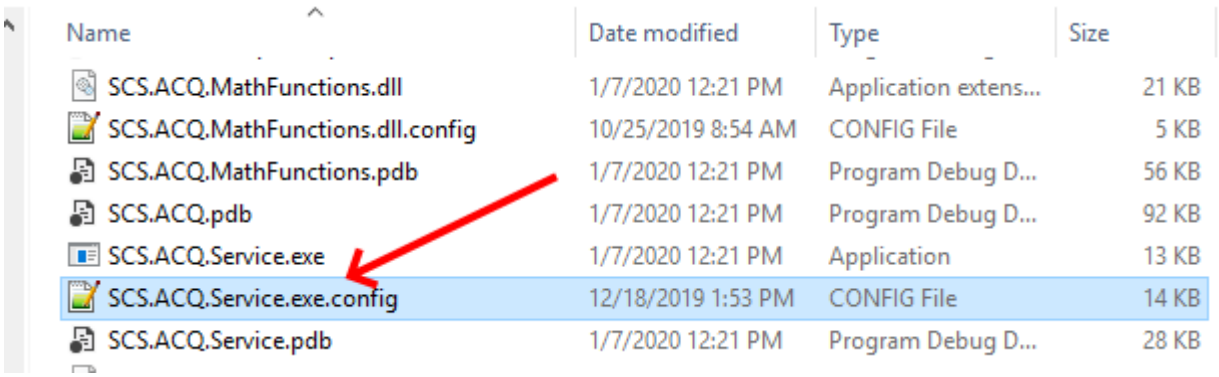
This is beyond the scope of the help manual, however setting up your own certificate authority is an option. In the vast majority of cases this will be to complicated a solution unless you already have one in place for other reasons. If so you may and should take advantage of it for SCS.

How to switch all SCS services to use the HTTP protocol instead of the HTTPS

Be default, all SCS services use the HTTPS protocol when connecting and communicating with the website. If you are disabling HTTPS or do not have a valid certificate you should let each service know that it should be using HTTP instead of HTTPS or you will introduce performance and functionality issues.

To do this you must browse to the directory you installed SCS into (normally D:\SCS) and open the Services directory. Inside there each SCS service will have it's own folder which contains most of it's execution code and configuration files.

Inside each folder you must find the service's configuration file (name follows this syntax: *SCS.[Service Name].Service.exe.config*) and edit it (Administrative mode may be required to edit and save this file).



Once the file is open search for the <appSettings> node. Inside there you will find a key named [Service].HubBaseUrl. This is the setting you must change, all you have to do is remove the 's' from https (eg transform it from https://.... to http://....). Do not change anything else in the file nor any other part of that line.

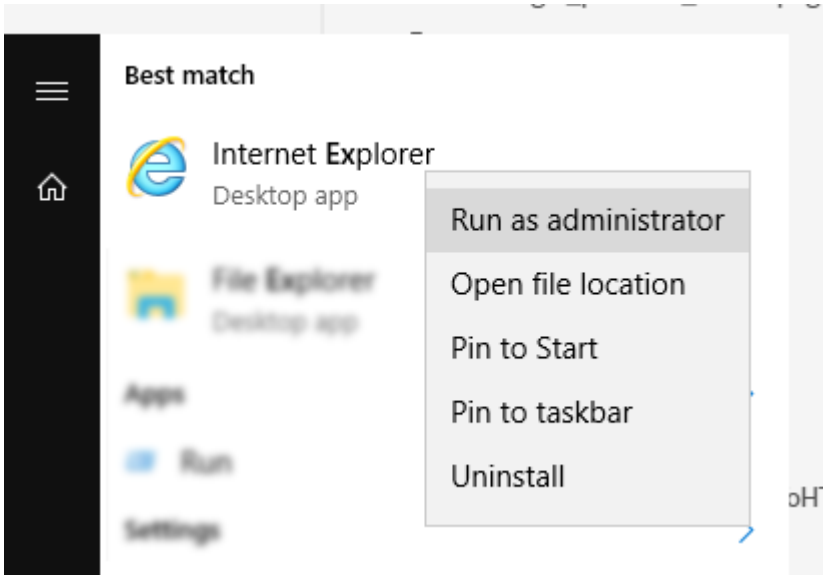
```
</startup>
<appSettings>
  <add key="ACQ.HubBaseUrl" value="https://localhost:44389" />
  <add key="ClientSettingsProvider.ServiceUri" value="" />
  <add key="BulkLoadMessageBufferSize" value="10" />
  <add key="OutputFileNameFormat" value="3" />
</appSettings>
<runtime>
```

Save the file and proceed onto the next service until all are modified.

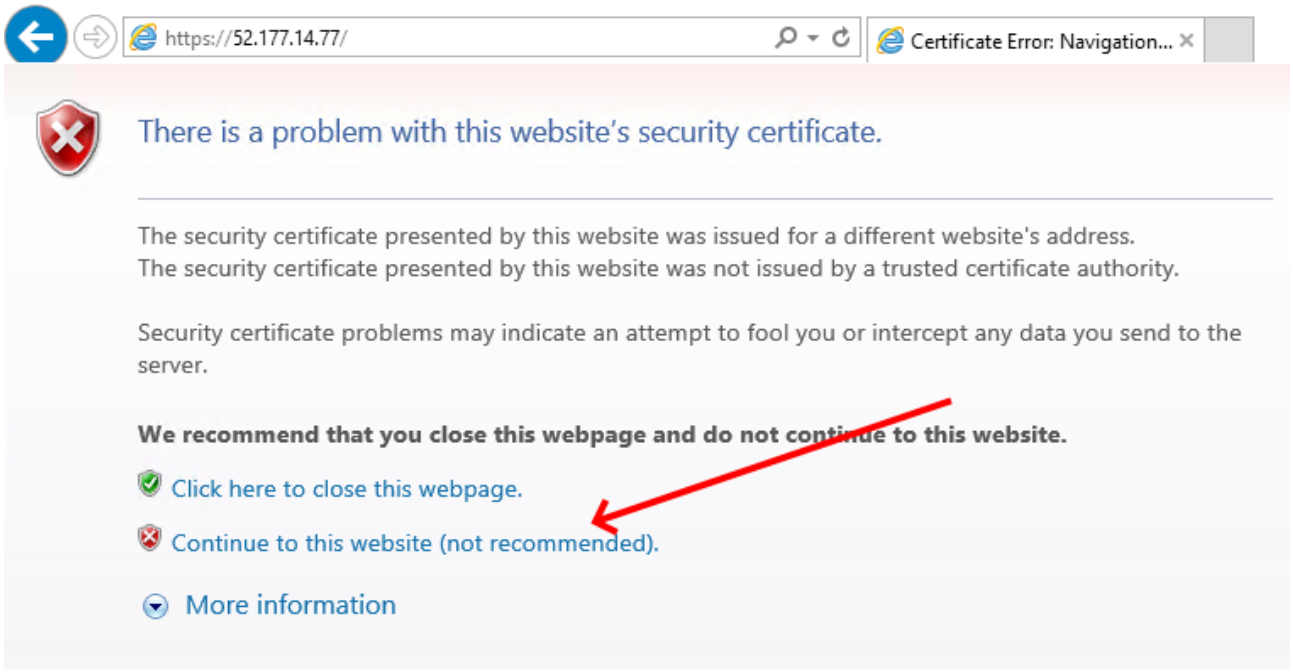
Trust a self-signed certificate

On each remote computer that you wish to operate SCS on you must:

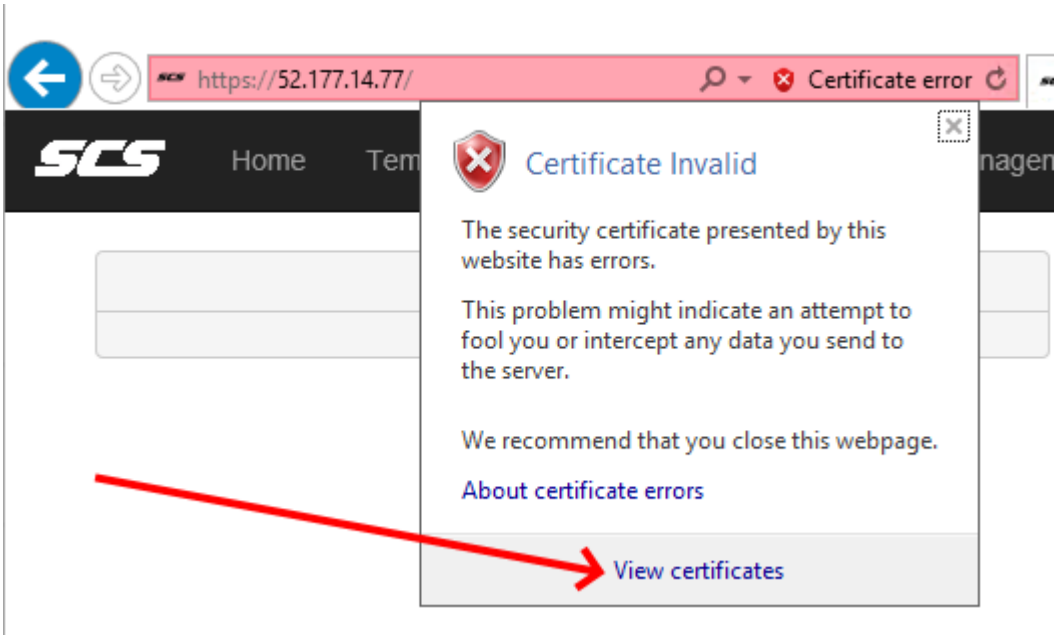
Open IE as an Administrator



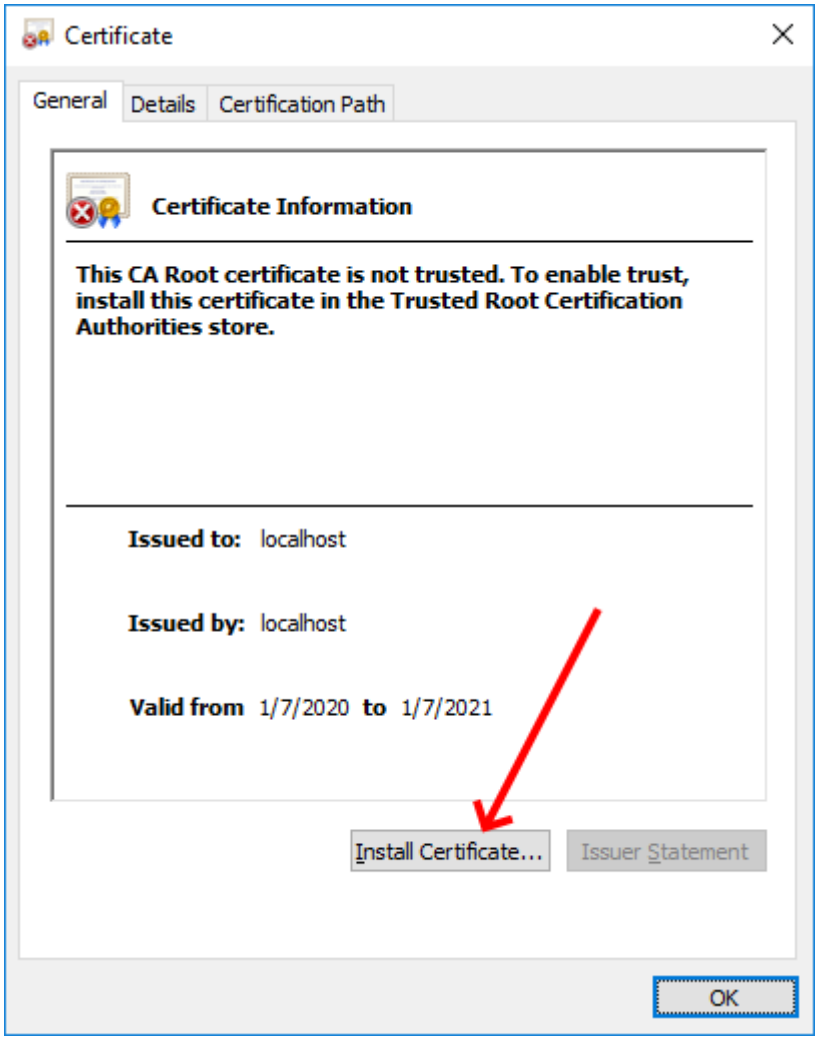
Browse to the SCS website in that browser, accept the certificate warning and continue on to the website



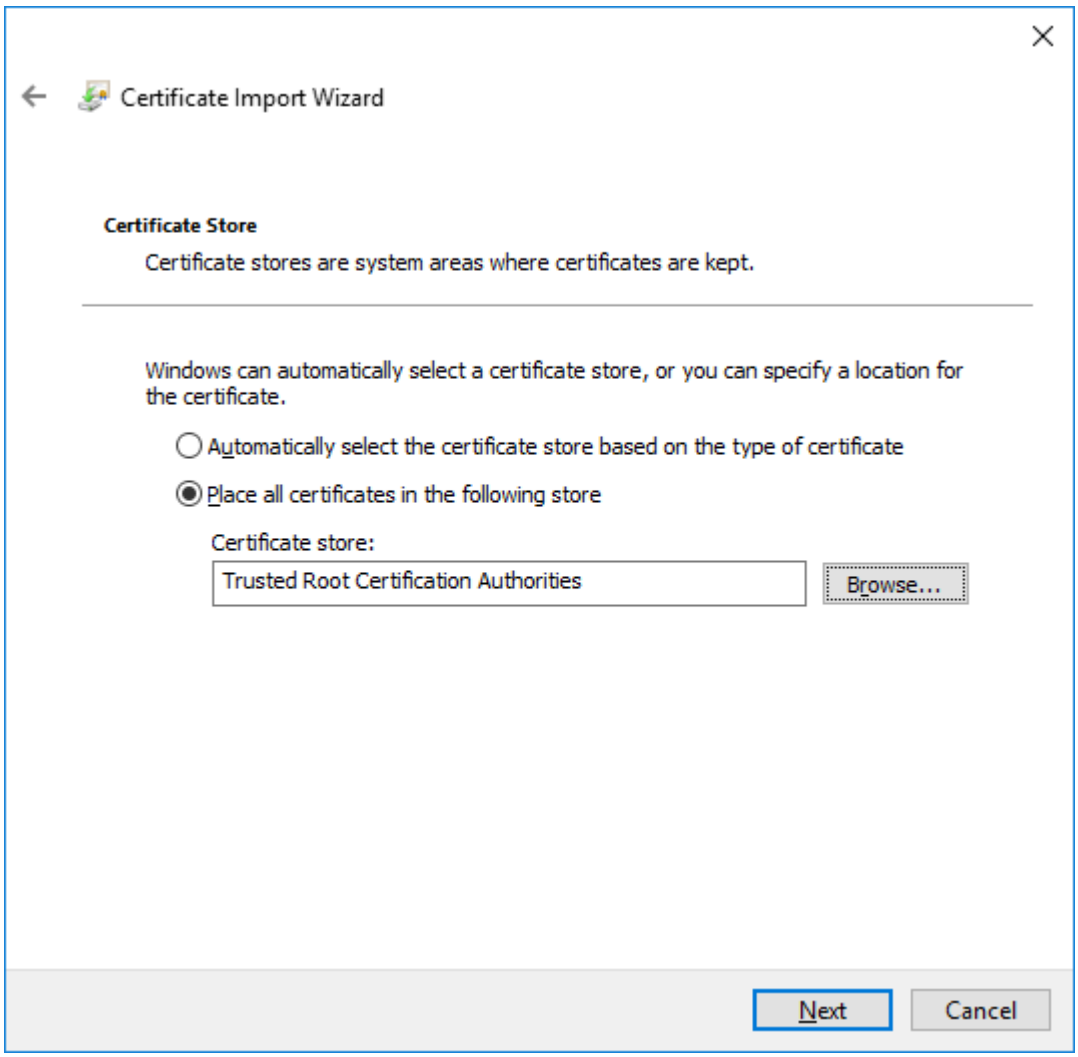
On the SCS site you'll notice the bar is red, click the Certificate Error text and View certificates



Click Install Certificate (note if this is grayed out or not there you are most likely not running IE in admin mode)



Choose Local Machine and put it in the Trusted Root Certification Authorities container



Now when you browse to the SCS server it should not present the security warnings any longer.

The primary purpose of SCS is to collect data from various sensors and log their outputs to persistent storage. Data visualization, packaging, marshaling, dissemination, QA/QC, submission and all the other functionality SCS provide are rendered pointless if no sensor data is being acquired. That being the case, it can be argued the two most important sub-systems inside SCS are CFE (covered in this section) and ACQ. CFE is responsible for defining the what and how of data acquisition, ACQ is responsible for the execution of the plan generated by CFE and the actual logging of the described data. Both applications are tightly coupled with the business rules of the database to provide consistent and quality sensor configuration while providing extensibility of the history of sensor devices over time

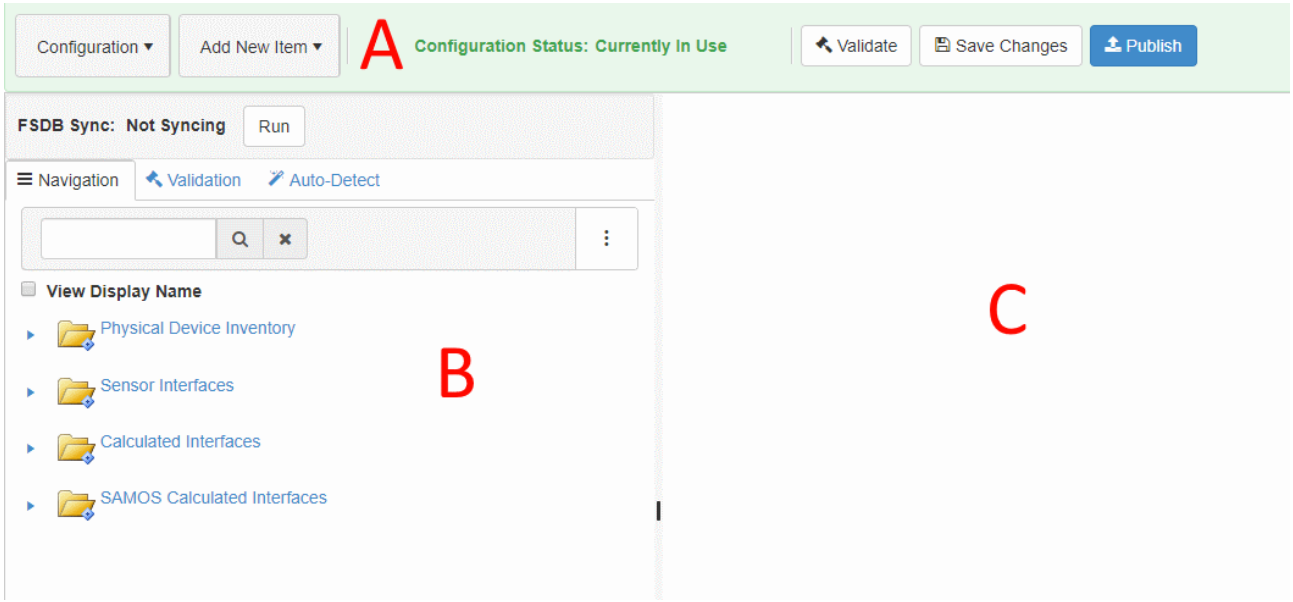
CFE is the location where SCS administrators can add, remove or update the suite of sensor feeds that SCS is acquiring. This is were you keep track of your physical inventory and their logical ingestion rules inside SCS. This is also where you can create custom derived data (such as converting a temperature from Celsius to Fahrenheit or converting a voltage reading to a more useful scientific value).

CFE is critical to the successful setup and operation of SCS and is restricted to administrative users only. In addition it can only be operated/opened on a single workstation at a time to avoid any possibility of mis-matched sensor configurations and reduce confusion.

Interface Overview

CFE can be accessed directly from the main menu once you have logged in, assuming you have administrator rights.

The user interface consists of a primary command toolbar (A) at the top which has a variety of dropdown lists, buttons and even color indicators to help relay the state of your configuration. Below the toolbar you will find the core of the sensor configuration tooling with a tab strip and tree on the left (B) and a large editing pane on the right (C). Each of these sections is described in more detail in this guide.



Configuration States

CFE acts somewhat independently of all the other systems and clients in SCS. It can be moderately complex and in fact the majority of the database is dedicated to this application. It maintains and manages two basic configuration states:

Working Configuration: The sandbox configuration being edited and updated via CFE

Published Configuration: The Live/Production configuration being consumed and used by all active SCS subsystems and clients.

You start off with a Working Configuration, this is your sandbox to play and design in. When you add, delete and update sensors, messages, devices, etc this is all being done in the working configuration. Once you are happy with your product you *Publish* the Working Configuration which updates the Published Configuration to reflect your changes.

Fleet Sensor Database

The FSDB synchronization status is displayed in the CFE UI. This sync process occurs automatically at regular intervals behind the scenes via the FSDB service. However, if you wish to manually initiate a synchronization task you can click the *Run* button and one will immediately start.

Navigation Tab

The navigation tab (B) is your main organizational structure in CFE. There is a search box above the tree in which you can search for keywords or text throughout the configuration. A result list will appear displaying all your matches, click the button to immediately goto the element of interest in the tree.

there is a set of buttons which also allow you to automatically expand or collapse the entire tree to a certain level. By clicking the various levels (1 - completely collapsed, 5 - completely expanded) you can perform a mass visualization operation on all nodes of the tree.

You can choose to display items by their system generated names, or the user-assigned *Display Names* via the *View Display Name* checkbox located at the top of the tree.

The tree itself displays all your physical inventory, sensor interfaces, calculated interfaces and SAMOS interfaces in a logical form. Expanding any node and clicking any item will load it's data (for editing / review) in the details pane (C) on the right.

Why does SCS care about Physical Inventory?

A *Physical Device* describes the kinds of actual devices (i.e. electronic equipment) that may be interfaced to the system. The database encapsulates all of the information about the physical sensor including the make, model, installation and calibrations details. Aside from helping to promote and advertise the overall sensor capabilities of your ship this metadata plays an important role once your data comes to shore. Metadata is used to verify accuracy and validate the integrity of the sensor data collected. It is also useful for generating statistical correlations between various data sets. If you contribute to SAMOS you already know of the importance the science parties on shore place on up to date and complete metadata regarding the ship and sensors used. Do not forget to think long term as well, the scientists of the future who will use your data also might have as-yet unknown requirements from which this type of metadata may play a critical role.

Metadata Standards

Metadata or 'the documentation of data' serves the purpose of making data discoverable, usable and understandable. A variety of metadata standards and formats have been developed over time to support data discovery and data documentation.

Metadata can be represented in a number of standards and formats. Many discipline-specific or community-specific metadata standards have been developed to support data management and data discovery systems and to capture and convey information to users. Examples include Directory Interchange Format (DIF), Ecological Metadata Language (EML), Sensor Model Language (SensorML), Climate Science Modeling Language (CSML), and NetCDF Markup Language (NcML). One commonly used metadata standard, the Federal Geographic Data Committee's (FGDC) Content Standard for Digital Geospatial Metadata (CSDGM) was developed in support of the coordinated development, use, sharing, and dissemination of geospatial data on a national basis.

Additionally, the International Organization for Standardization (ISO) developed a series of standards to describe geographic information. The use of metadata to document environmental data is described in NOAA Administrative Order 212-15 which outlines the Management of Environmental Data and Information. NOAA's Environmental Data Management Committee's (EDMC) Data Documentation Planning Directive "establishes ISO 19115 Parts 1 and 2 and a recommended representation standard (ISO 19139) for documenting NOAA's environmental data and information."

Metadata Creation

The production, validation, and management of metadata records can be challenging and time consuming. A number of content sources, formats, storage structures and standards exist and change over time. Metadata creation is often further impeded by complex of metadata standards and a variety of available tool-sets. Some methods for automating metadata transformations are described in XML Transformations.

Metadata Training

The National Centers for Environmental Information have personnel who are experienced working with data providers to develop metadata for their data. Information about NCEI's current online metadata training courses can be found on the NCEI Metadata Training Page.

Workbooks have been developed as a guide to implementing the ISO 191** metadata standards and are available in PDF format below:

- ISO 19115:2003 Geographic Information - Metadata Workbook (2.5 MB) - Guide to Implementing ISO 19115:2003(E), the North American Profile (NAP), and ISO 19110 Feature Catalogue.
- ISO 19115:2003 Geographic Information - Metadata - Biological Extensions Workbook (2.75 MB) - Guide to Implementing ISO 19115:2003(E), the North American Profile (NAP), and ISO 19110 Feature Catalogue with Biological Extensions.
- ISO 19115-2:2009 Geographic Information - Metadata - Part 2: Extensions for imagery and gridded data Workbook (2.99 MB) - Guide to Implementing ISO 19115-2:2009(E), the North American Profile (NAP), and ISO 19110 Feature Catalogue.

- https://www.ncddc.noaa.gov/metadata-standards/

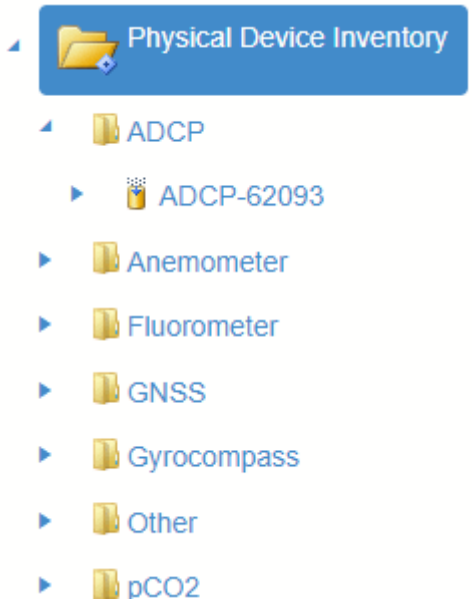
Maintaining your Physical Inventory

As detailed in the background, keeping your physical inventory up to date is extremely important in terms of metadata and the usefulness of your datasets after they have been archived. As keeping your physical

inventory up to date does not have any noticeable effect on the datasets being collected in real time it is tempting to ignore this completely, please try to avoid the temptation!

OMAO has attempted to compile a list covering categories of physical sensing devices and sensors that would be found on a typical research vessel. This list tries to be holistic based off the common vocabularies found in various fleets (mainly UNOLS) and international standards. However, we cannot guarantee this is a complete set nor that it is/should be any standard, it is just our take on the current sensors out there.

ADCP	Acoustic Doppler Current Profiler - a SONAR that measures water current velocities over a depth range.
Anemometer	Measures wind speed and direction.
CTD	An integrated hydro system which measures conductivity, temp, pressure, etc.
Expendable Probe	A hand/deck-launched single-use probe - XBT, XCTD, XSV, XCP, etc.
Flowmeter	Measures rate of water flow - mechanical, optical, electromagnetic, etc.
Fluorometer	Measures fluorescence (usually for phytoplankton).
GNSS	Global Navigation Satellite System - GPS/WAAS, GLONASS, Galileo, etc.
Gravimeter	Measures the Earth's local gravitational field.
Gyrocompass	Compass with a motorized gyroscope that tracks true north (heading).
HDSS	Hydrographic Doppler Sonar System
INS	Determines spatial position and motion using inertial sensors with input from additional sensors (eg. satellite fixes).
Magnetometer	Measures the strength and/or direction of the Earth's magnetic field.
Metstation	Integrated meteo system measures temp, pressure, humidity, etc.
Multibeam	Multiple Formed Beam Echosounder (mapping sonar).
Other	A device type that is not found in the list. Please enter the type in the "Other Type" field.
pCO2	Measures partial pressure of dissolved carbon dioxide.
PTU	Measures any/all of barometric pressure, air temperature, and relative humidity.
Radiometer	Measures radiation - pyranometer, pyrheliometer, pyrgeometer, etc.
Rain Gauge	Measures the amount of liquid precipitation (udometer).
Singlebeam	Single, dual, or split beam echosounder (profiling sonar); may be either fixed frequency or chirped.
Speed Log	Measures Doppler near surface vessel speed through water.
SSV	Sea Surface Sound Velocimeter - typically used as input to multibeam.
Thermometer	Measures air or water temperature.
Time Server	Reads time from reference clock (GPS, WWVB, etc.) and distributes locally.
Transmissometer	Measures fraction of light absorbed or scattered by particles in water.
TSG	Thermosalinograph - measures sea surface temp, salinity, etc. near seawater intake.
Underwater Positioning System	A system for the tracking and navigation of underwater vehicles or divers by means of acoustic distance and/or direction measurements, and subsequent position triangulation.



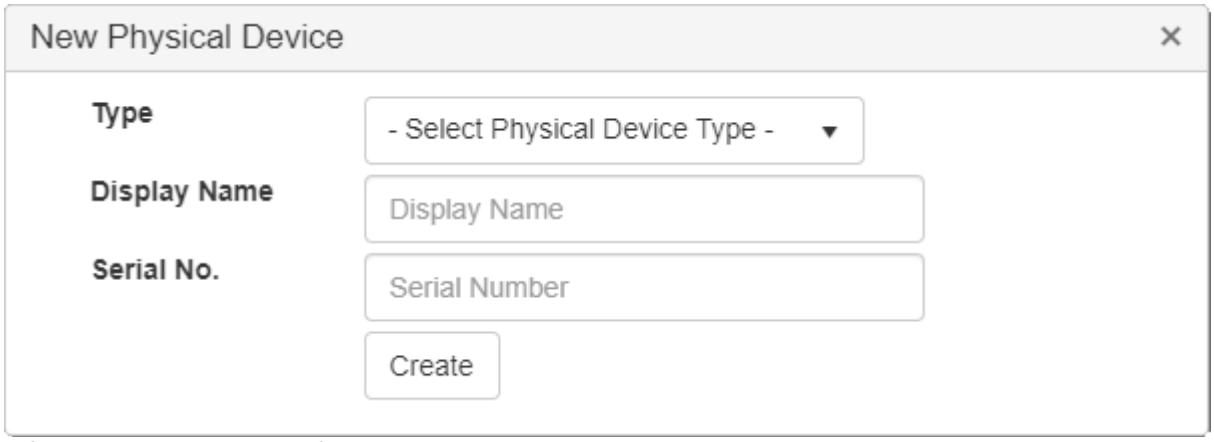
When you expand the *Physical Device Inventory* node in CFE it will group all Physical Devices you have setup into the category you specified they belong to.

If a category seems to be missing it is because you have not yet created/assigned a Physical Device to it yet.

Be aware, since our list is subject to change we provide the ability for you, as a user, to classify a sensor in the ***Other*** category. If you do so that category will be presented to us for inclusion into the master set and then will be available for all to use.

Creating a Physical Device

To add a new Physical Device you can click the *Physical Device* link under the *Add New Item* dropdown in the main toolbar, or simply right click on the *Physical Device Inventory* tree node and choose *Create Physical Device*.



As with other items in CFE a base set of information is required before the object can be created.

For Physical Devices you must supply the type, a display name and a serial number.

Once you have supplied the basics above CFE will automatically add your new item to the appropriate place in the tree and select it for further editing. The detailed pane will now contain a new and mostly empty form. Fill out the remaining information the best that you can, any required fields will be checked upon validation if missing or in error. Don't forget to Save your changes!

Editing

As with other CFE items, a Physical Device details pane has two tabs, a *Basic* and an *Advanced*.

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - For a more 'user friendly' name edit the Display Name attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique (defaults to System Name).

- Device Type
  - The Type of Physical Device being defined.
  - Changing this will have a domino effect such as moving it in the *Physical Inventory* tree, changing the default values for *Manufacturer*, etc.
- Manufacturer
  - The party responsible for creating of this Physical Device
- Model No
  - The model of the instrument
- Serial No
  - The manufacturer supplied serial number of the instrument
- Location
  - The general location of the instrument (exact location can be specified in the *Advanced* tab if known)
- Not Calibrated
  - If checked it is noted that this device is not able to be calibrated.
  - If unchecked then this device should be periodically calibrated to keep its data scientific grade
- Latest Calibration
  - If the device is able to be calibrated, this is where you can keep a record of said calibrations
  - You are able to specify when the calibration took place, add any notes of interest and actually upload any related documentation (such as the cal sheets, certs, etc).
- Latest Test
  - If the device has been tested it can be noted here along with who conducted the test and what results they came up with.
- Installed To
  - If the Physical Device is actually supplying data to SCS you can link those datasets to this specific device by *Installing* the device to the Sensor Interface. When you swap a physical sensor out be sure to update this portion of SCS to reflect your changes!

The *Advanced* tab consists of the following fields:

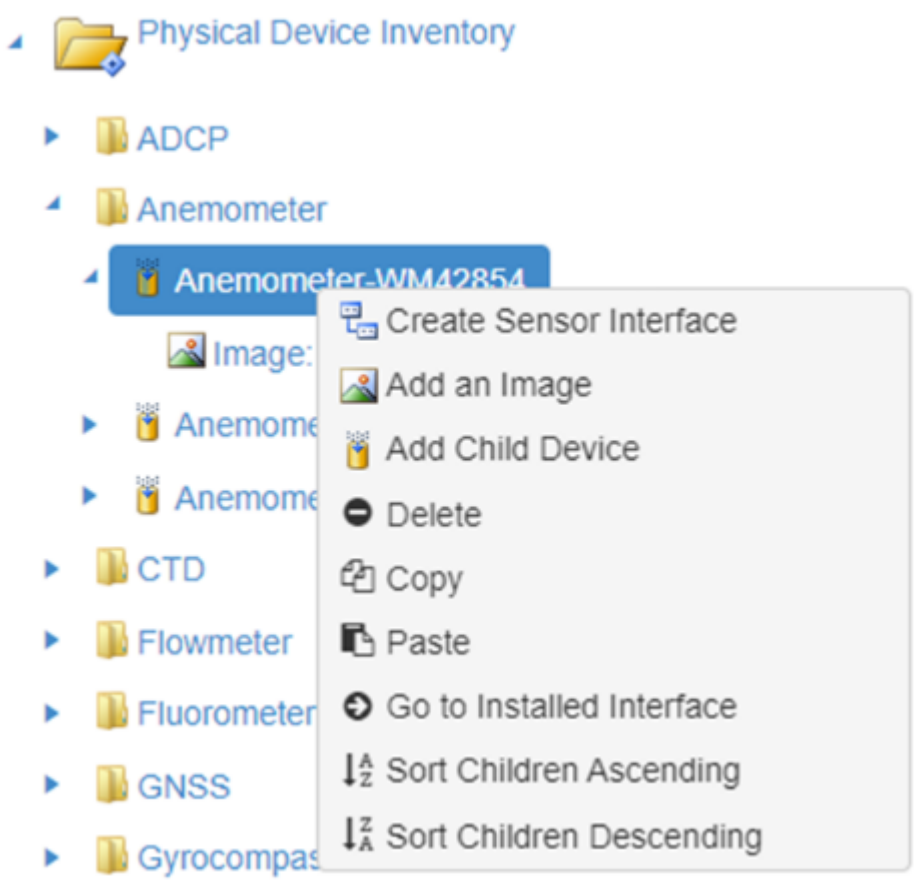
- Comment
  - A general comment pertaining to this item
- Keywords
  - Optional tags used to assist with text searches and metadata
- Location Measurement Reference
  - In the event you have a detailed surveyed location for this device this would be the origin point the XYZ offset would be based off of.
- Survey Location
  - The surveyed location relative the the *Location Measurement Reference* where this device is physically located.
- CD No
  - Sunflower / NOAA inventory ID
- Reference Device Order
  - Sets the priority this device's data streams have relative to other instances in the same category. For instance, this is how you define your primary GPS vs your secondary

GPS, etc. The lower the number the more preferential it's data feed is (eg your best GPS should be 0, your next best should be 1, etc).
    - Drag and drop (first column) to change the preferred order.
    - In the event two or more devices have the same *reference order* they are treated as equals.
    - This is how SCS determines which data source to use when client templates include *Reference* sourced data. For instance, instead of explicitly selecting a Latitude and Longitude for an event, you can listen to the *reference* Lat/Lon. SCS will automatically provide the best GPS data (based upon *Reference Device Order*) to the event. If the primary GPS goes down or QA/QC starts flagging it then SCS will automatically fallback to the next device in the list and continue to supply the best data possible to the event.
    - This property is ignored if the Physical Device is not installed to a Sensor Interface.
- Time Source
    - The source of time used by this device (GNSS devices, NTP servers, local system clock, etc).
- Data External to SCS
    - If checked then this Physical Device is for metadata reference only. It may be a self-contained system, a backup device sitting on a shelf in the closet (spare GPS) or even being ingested by another system (eg multi-beam, etc). Even if not part of SCS it can be described here to be included in the overall vessel sensor capabilities as reported and advertised on shore.
    - If unchecked the data should be being actively collected by SCS
- Installed To
    - Similar to the *Installed* property on the *Basic* tab, however this instance allows you to manage historic installations.

Removing

To remove a Physical Device simply right click on it in the tree list and choose the *Delete* option.

Additional Operations / Context Menu



Right clicking in the *Physical Device Inventory* on a category or Physical Device also presents a variety of operations you can execute. Some are only relevant if conditions are met (eg you cannot *paste* if you have not yet *copied* something), most are self explanatory.

You can create a new sensor interface linked to this device.

You can add images associated with the device.

You can build a hierarchy of children (sub-devices) for the device.

You can delete, copy or paste entire devices.

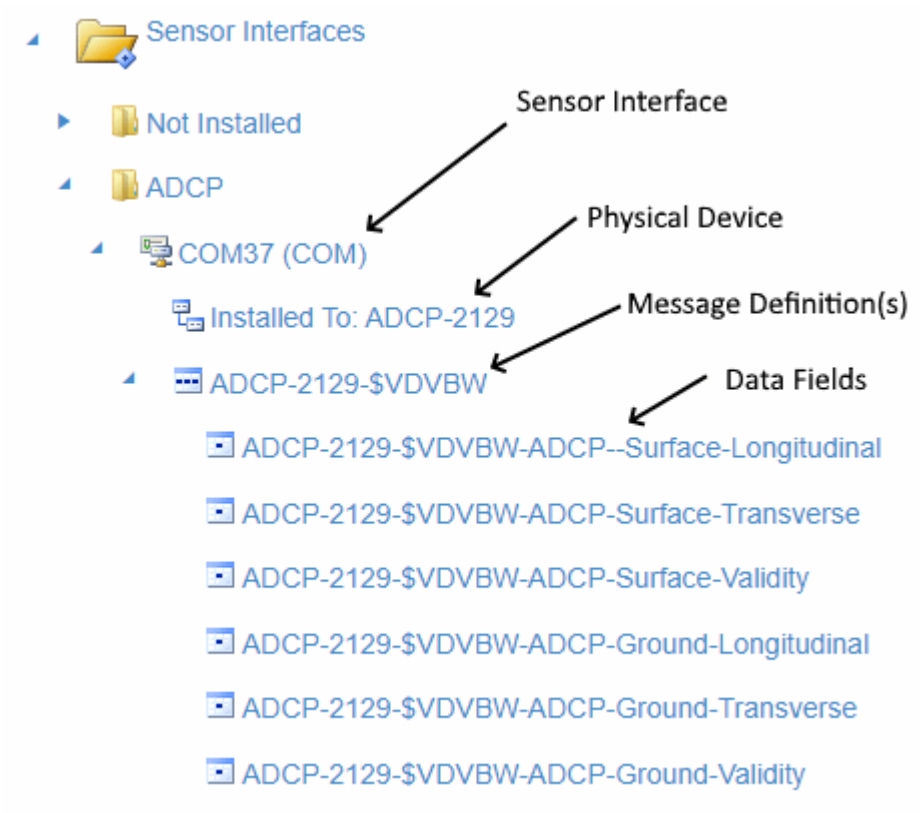
You can immediately go to the Sensor Interface (if any) consuming the data from this device.

You can also sort children (items appearing inside the node) alphabetically.

Sensor Interfaces

The primary purpose of CFE is to provide you with an interface to create, edit and remove sensor data feeds inside SCS. When a new sensor is brought online you can describe it physically via the Physical Devices portion of CFE, but in order for SCS to actually start ingesting, recording and disseminating the data being provided from the sensor you must also add it to the *Sensor Interfaces* portion of CFE.

The Sensor Interface portion of the tree breaks each interface down in a hierarchy, each level of this hierarchy is described in the following paragraphs.



If your interface is linked to a physical device then it will be grouped inside the 'type' of physical device to help with organization.

In this instance you can see we have a *Sensor Interface* defining the data coming in on COM port 37.

It is defining the data coming from the ADCP-2129 *Physical Device*.

It has a single Message Definition describing the $VDVBW data

It breaks the $VDVBW data stream into six (6) Data Fields.

Interface Level

A *Sensor Interface* encapsulates the interface of a physical device to ACQ. You can also think of it as describing the source of a physical device's messages, without knowing anything about the physical device itself. A Sensor Interface has parameters such as the baud rate, TCP/IP port and others that further describe the interface.

SCS has four (4) Sensor Interface types, each one distinct in how it is interfaced to its data source.

1. COM: Connects to its physical device via a COM port using the RS-232 protocol. A COM device receives messages without prompting the physical device.
2. Polled COM: Same as a COM device, except the physical device sends a message only when ACQ prompts it to.
3. Network: Connects to a physical device over a network using a TCP or UDP protocol.
4. Manual: The physical device is a keyboard that passes data to ACQ over a TCP connection. The Manual Interfaces SCS Widget allows the user to send observational data from their keyboard to ACQ for logging, display, etc.

When you configure Sensor Interfaces, you are specifying the following:

- What sensor equipment is connected to SCS
- The source of the sensor data (COM port, network socket, keyboard entry, etc.)
- What messages the sensors are supplying
- How each message is divided into data fields
- How the sensor data will be displayed in real time
- How SCS will log the data to disk.

Message Definition Level

Each Sensor Interface may deliver multiple distinct messages that are transmitted from the Physical sensor. Message Definitions define all of the information needed to acquire a specific message from a physical sensor. A sensor may transmit multiple messages with each being described separately. Message Definitions are sometimes closely coupled with the kind of Sensor Devices and other times Message Definitions are re-used by multiple Sensor Interfaces.

> A Sensor Interface may contain multiple Message Definitions such as a GPS receiver that may be sending multiple NMEA messages (GGA, VTG, etc.)

SCS has three (3) main types of Message Definitions.

1. Delimited: The data fields inside the RAW stream are delimited via one or more ASCII characters (comma, tab, etc).
2. Fixed: The data fields inside the RAW stream are defined by hard and fixed character positions.
3. NMEA: A special implementation of Delimited, message and data fields comply with the NMEA-0183 standard.

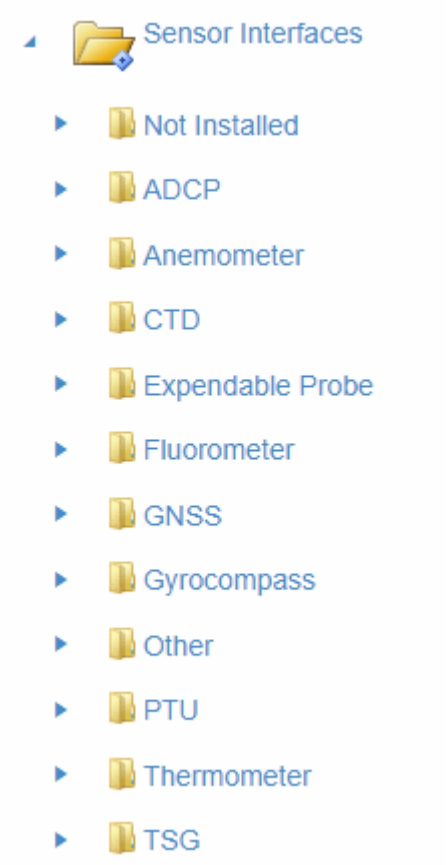
Data Field Level

Every Sensor Interface passes one or more Message Definitions to ACQ. Each message in turn has one or more Data Fields associated with it, one for each measurement that the physical device makes. Each data field has parameters that establish how to decode and monitor one measurement in the message. These parameters specify such things as where the measurement is in the message, what units it has and what type of measurement it is.

Once ACQ isolates a Data Field it can pass on the value to other SCS apps that can display them, pass them on for derived sensor calculations, or use them in other ways described in later chapters. Since serial messages have fixed format, serial data fields are delimited by their start and end character positions in the message. NMEA Messages are variable length where the data fields are delimited by a comma. NMEA data fields may also be attributed special features of translation to allow a code embedded in the message to be translated to a more meaningful value.

Maintaining your Sensor Interfaces

The main purpose of CFE is to configure and maintain the collection of Sensor Interfaces and data streams that ACQ and other SCS components read, store, analyze and display.



Each Sensor Interface is grouped into the type of Physical Device associated (Installed) with it.

If you have not associated the interface with an actual sensor then it will be automatically placed in the *Not Installed* tree node.

Expanding any category of Physical Device will display all Sensor Interfaces providing data from those devices.

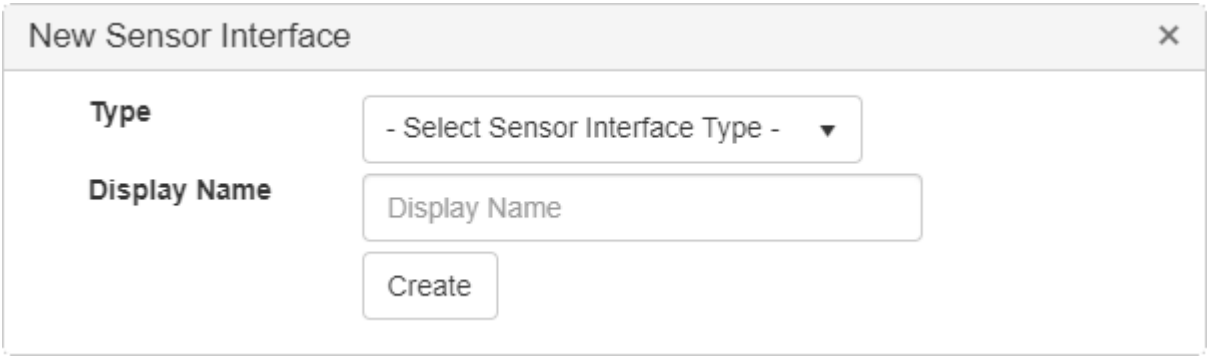
Expanding any Sensor Interface will display all Message Definitions being provided over that interface.

Expanding any Message Definition will display all the Data Fields defined inside that definition.

The above hierarchy is defined in more detail in the background portion of this section of help.

Creating Interfaces

To add a new Sensor Interface you can click the ***Sensor Interface*** link under the *Add New Item* dropdown in the main toolbar, or simply right click on the *Sensor Interfaces* tree node and choose *Create Sensor Interface*.



As with other items in CFE a base set of information is required before the object can be created.

For Sensor Interfaces you must supply the type, a display name and, depending on which type you select, you may have a few more data items to fill out to further define your interface before proceeding.

Once you have supplied the basics above CFE will automatically add your new item to the appropriate place in the tree and select it for further editing. The detailed pane will now contain a new and mostly empty form. Fill out the remaining information the best that you can, any required fields will be highlighted for you. Don't forget to Save your changes!

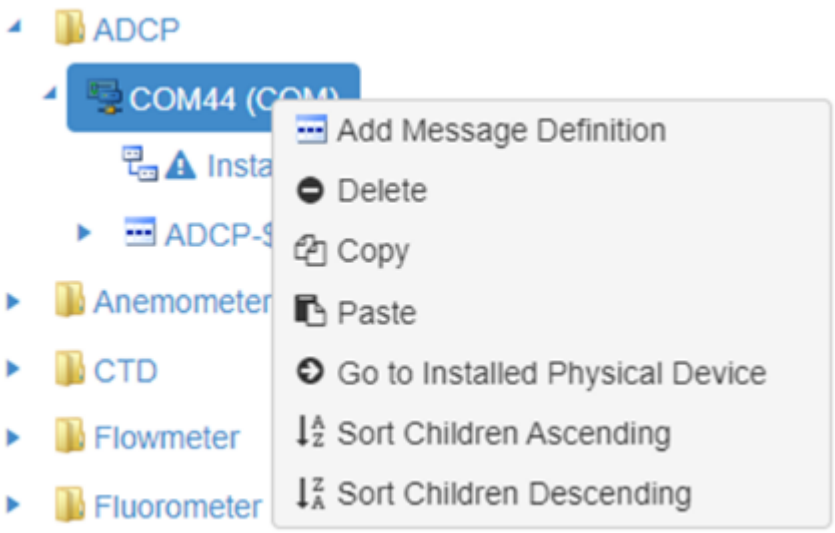
Editing

As with other CFE items, a Sensor Interface details pane has two tabs, a *Basic* and an *Advanced*. Depending on which type of interface you are editing you will have a different set of items to edit. Please consult the help page matching the selected interface type for more information.

Removing

To remove a Sensor Interface simply right click on it in the tree list and choose the *Delete* option.

Additional Operations / Context Menu



Right clicking in the *Sensor **Interfaces*** on a category or Sensor Interface also presents a variety of operations you can execute. Some are only relevant if conditions are met (eg you cannot *paste* if you have not yet *copied* something), most are self explanatory.

You can build out the Message Definitions for the interface.

You can delete, copy or paste entire interfaces

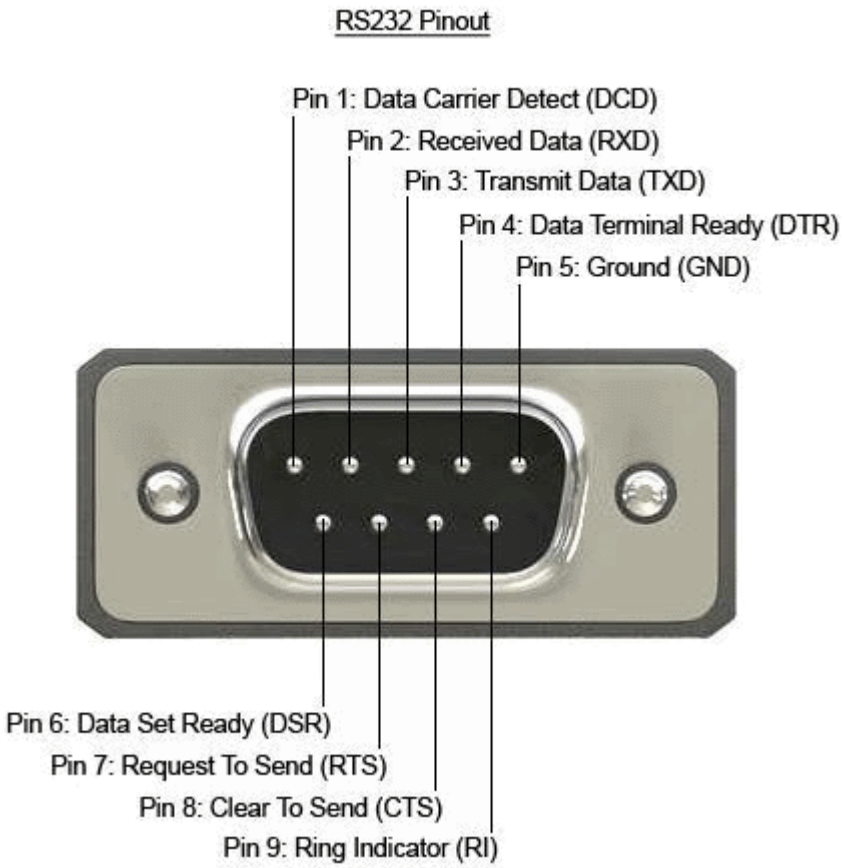
You can immediately go to the Physical Device (if any) currently providing the data through this interface.

COM Interfaces

COM (Communication port) is the original, yet still common, name of the serial port interface on IBM PC-compatible computers.

Most PC-compatible computers in the 1980s and 1990s had one or two COM ports. As of 2007, most computers shipped with one or no physical COM ports. As of 2014, consumer-grade PC-compatible computers don't include any COM ports, though some of them do still include a COM header on the motherboard.

After the RS-232 COM port was removed from most consumer-grade computers in the 2000s, an external USB-to-UART serial adapter cable was used to compensate for the loss. A major supplier of these chips is FTDI.



The COM ports are interfaced by an integrated circuit such as 16550 UART. This IC has seven internal 8-bit registers which hold information and configuration data about which data is to be sent or was received, the baud rate, interrupt configuration and more. In the case of COM1, these registers can be accessed by writing to or reading from the I/O addresses 0x3F8 to 0x3FF.

If the CPU, for example, wants to send information out on COM1, it writes to I/O port 0x3F8, as this I/O port is "connected" to the UART IC register which holds the information that is to be sent out.

The COM ports in PC-compatible are typically defined as:

- COM1: I/O port 0x3F8, IRQ 4
- COM2: I/O port 0x2F8, IRQ 3
- COM3: I/O port 0x3E8, IRQ 4
- COM4: I/O port 0x2E8, IRQ 3

Given the lack of COM support in modern computers and the need for a large number of serial inputs required by SCS most ships utilize an multi-port adapter to provide a series of external COM Interfaces. Most NOAA ships use COMTROL or DIGI adapters to provide these inputs.

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the *Display Name* attribute below
- Display Name
  - User supplied name

    - Can be anything you wish though must be unique.
- Type
    - The Type of Sensor Interface being defined.
    - This is set upon creation and cannot be changed after the fact. You can copy the interface's Message Definitions and paste them into a new interface of a different type if needed.
- Enabled
    - A flag letting SCS know if this interface is actively providing data or if it should be ignored by client applications such as ACQ.
- Installed To
    - If the Physical Device is actually supplying data to SCS you can link those datasets to this specific device by *Installing* the device to the Sensor Interface. When you swap a physical sensor out be sure to update this portion of SCS to reflect your changes!
- COM Port
    - The actual numeric port SCS should open to pull data from this interface.
- Baud Rate
    - The rate at which data should be expected to flow through this interface.

The *Advanced* tab consists of the following fields:

- Comment
    - A general comment pertaining to this item
- Keywords
    - Optional tags used to assist with text searches and metadata
- Installed To
    - Similar to the *Installed* property on the *Basic* tab, however this instance allows you to manage historic installations.
- Data Bits
    - The number of data bits in each character can be 5 (for Baudot code), 6 (rarely used), 7 (for true ASCII), 8 (for most kinds of data, as this size matches the size of a byte), or 9 (rarely used). 8 data bits are almost universally used in newer applications. 5 or 7 bits generally only make sense with older equipment such as teleprinters.
- Stop Bits
    - Stop bits sent at the end of every character allow the receiving signal hardware to detect the end of a character and to resynchronise with the character stream. Electronic devices usually use one stop bit. If slow electromechanical teleprinters are used, one-and-one half or two stop bits are required.
- Parity
    - A method of detecting errors in transmission. When parity is used with a serial port, an extra data bit is sent with each data character, arranged so that the number of 1 bits in each character, including the parity bit, is always odd or always even. If a byte is received with the wrong number of 1s, then it must have been corrupted. However, an even number of errors can pass the parity check.

      Electromechanical teleprinters were arranged to print a special character when received data contained a parity error, to allow detection of messages damaged by line noise. A single parity bit does not allow implementation of error correction on each character, and communication protocols working over serial data links will have higher-level mechanisms to ensure data validity and request retransmission of data that has been incorrectly received.

The parity bit in each character can be set to one of the following:

- None (N) means that no parity bit is sent at all.
- Odd (O) means that parity bit is set so that the number of "logical ones" must be odd.
- Even (E) means that parity bit is set so that the number of "logical ones" must be even.
- Mark (M) parity means that the parity bit is always set to the mark signal condition (logical 1).
- Space (S) parity always sends the parity bit in the space signal condition (logical 0).

- Aside from uncommon applications that use the last bit (usually the 9th) for some form of addressing or special signaling, mark or space parity is uncommon, as it adds no error detection information. Odd parity is more useful than even parity since it ensures that at least one state transition occurs in each character, which makes it more reliable at detecting errors like those that could be caused by serial port speed mismatches. The most common parity setting, however, is "none", with error detection handled by a communication protocol.

- Flow Control
  - In many circumstances a transmitter might be able to send data faster than the receiver is able to process it. To cope with this, serial lines often incorporate a "handshaking" method, usually distinguished between *hardware* and *software* handshaking.

    Hardware handshaking is done with extra signals, often the RS-232 RTS/CTS or DTR/DSR signal circuits. Generally, the RTS and CTS are turned off and on from alternate ends to control data flow, for instance when a buffer is almost full. DTR and DSR are usually on all the time and, per the RS-232 standard and its successors, are used to signal from each end that the other equipment is actually present and powered-up. However, manufacturers have over the years built many devices that implemented non-standard variations on the standard, for example, printers that use DTR as flow control.

    Software handshaking is done for example with ASCII control characters XON/XOFF to control the flow of data. The XON and XOFF characters are sent by the receiver to the sender to control when the sender will send data, that is, these characters go in the opposite direction to the data being sent. The circuit starts in the "sending allowed" state. When the receiver's buffers approach capacity, the receiver sends the XOFF character to tell the sender to stop sending data. Later, after the receiver has emptied its buffers, it sends an XON character to tell the sender to resume transmission. It is an example of in-band signaling, where control information is sent over the same channel as its data.

    The advantage of hardware handshaking is that it can be extremely fast; it doesn't impose any particular meaning such as ASCII on the transferred data; and it is stateless. Its disadvantage is that it requires more hardware and cabling, and these must be compatible at both ends.

    The advantage of software handshaking is that it can be done with absent or incompatible hardware handshaking circuits and cabling. The disadvantage, common to all in-band control signaling, is that it introduces complexities in ensuring that a) control messages get through even when data messages are blocked, and b) data can never be mistaken for control signals. The former is normally dealt with by the operating system or device driver; the latter normally by ensuring that control codes are "escaped" (such as in the Kermit protocol) or omitted by design (such as in ANSI terminal control).

    If no handshaking is employed, an overrun receiver might simply fail to receive data from the transmitter. Approaches for preventing this include reducing the speed of the connection so that the receiver can always keep up; increasing the size of buffers so it can keep up averaged

over a longer time; using delays after time-consuming operations (e.g. in termcap) or employing a mechanism to resend data which has been corrupted (e.g. TCP).

The default setup for NMEA-0183 (not NMEA-0183HS) compliant sensors coming over serial COM ports is generally:

Baud Rate	4800
Data bits	8
Parity	None
Stop bits	1
Handshake	None

- Much of above was taken from: https://en.wikipedia.org/wiki/COM_(hardware_interface) and https://en.wikipedia.org/wiki/Serial_port

Manual Interfaces

Manual Interfaces provide a way for users to manually supply data to SCS and ACQ via their keyboard and 'manual' observations. In effect it allows the users themselves to act in the same manner as a sensor. These manual data items are generated and sent by users via the Manual Interface Widget.

Underneath the UI Manual Interfaces use the ship's network to listen for and process incoming data. As such they need a IP port to bind to on the server which clients can connect to.

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the *Display Name* attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique.
- Type
  - The Type of Sensor Interface being defined.
  - This is set upon creation and cannot be changed after the fact. You can copy the interface's Message Definitions and paste them into a new interface of a different type if needed.
- Enabled
  - A flag letting SCS know if this interface is actively providing data or if it should be ignored by client applications such as ACQ.

- Port
  - The port to use when establishing the socket connection (TCP) or to listen on (UDP)

The *Advanced* tab consists of the following fields:

- Comment
  - A general comment pertaining to this item
- Keywords
  - Optional tags used to assist with text searches and metadata

> ⚠ There is no option to 'install' a Manual Interface as it is not connected to any physical sensor. The concept being that a human is on the other end supplying the data which renders the installation and linkage portion of SCS between physical and logical interfaces as not applicable.

Network Interfaces

Network Interfaces in SCS send and receive traffic over the Internet Protocol (IP). It does these using either the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP). When creating a new Network Interface you must specify which protocol your sensor is using to output it's data and [optionally] the IP address of the sensor itself. Be sure the sensor traffic can reach the SCS server and isn't being blocked/dropped due to subnets, firewalls or other network setup and traffic control measures.

> A good way to test network sensors is to use a client based tool on the server itself (hyper-term, putty, wireshark, etc) to listen for the traffic and verify that it is arriving as expected.

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the *Display Name* attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique.
- Type
  - The Type of Sensor Interface being defined.
  - This is set upon creation and cannot be changed after the fact. You can copy the interface's Message Definitions and paste them into a new interface of a different type if needed.

- Enabled
  - A flag letting SCS know if this interface is actively providing data or if it should be ignored by client applications such as ACQ.
- Installed To
  - If the Physical Device is actually supplying data to SCS you can link those datasets to this specific device by *Installing* the device to the Sensor Interface. When you swap a physical sensor out be sure to update this portion of SCS to reflect your changes!
- Network Protocol
  - The IP protocol to use for this interface (must match method being used/expected by sensor itself)
- IP Address
  - The address of the sensor supplying the data to this interface
  - If sensor is providing data over UDP Broadcast then this should be set to 255.255.255.255.
- Port
  - The port to use when establishing the socket connection (TCP) or to listen on (UDP)

The *Advanced* tab consists of the following fields:

- Comment
  - A general comment pertaining to this item
- Keywords
  - Optional tags used to assist with text searches and metadata
- Installed To
  - Similar to the *Installed* property on the *Basic* tab, however this instance allows you to manage historic installations.

Polled COM Interfaces

Polled COM Ports are essentially the same as normal COM Ports with the exception that they send out a text prompt at regular intervals to 'poll' sensors reading and writing from the port. The data from the sensor is only output when a specific command is transmitted from SCS to the sensor. The message is either terminated by a special character or is defined at a fixed length.

Polled COM is a special implementation of the standard COM interface, therefore the majority of the settings and concepts are the same. However it does have a single additional property on the *Basic* tab:

- Polling Rate
  - The rate (in seconds) in which ACQ will output the prompt specified by the interface's Message Definitions.

Message Definitions under a Polled COM Interface also require a *Command Prompt* (the prompt ACQ will send out the COM port at the specified polling rate) and a *Queue Order* (the order which ACQ will serially send the various prompts, assuming there is more than 1 definition).

Delimited Definitions

In contrast to fixed messages, delimited messages may have variable record length. The order of the data fields is constant, but the number of characters in each field can change. To separate one Data Field from another delimiters are specified, the raw data is split based upon the one or more delimiters you specify.

> ⚠ As there is no *sentence label* to distinguish messages like in a NMEA Message Definition an interface cannot have more than one delimited message.

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the *Display Name* attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique.
- Type
  - The Type of Message Definition being defined.
  - This is set upon creation and cannot be changed after the fact.
- Log Messages
  - A flag lettings ACQ know if this stream should be logged to disk (files / database) or not.
  - If unchecked this data will still be available to all clients and calculated devices, but will not be stored on disk.
- Enabled
  - A flag letting SCS know if this interface is actively providing data or if it should be ignored by client applications such as ACQ.
- Termination Characters
  - The last character (often a control character) which signifies the end of the data string.

The *Advanced* tab consists of the following fields:

- Comment
  - A general comment pertaining to this item
- Keywords
  - Optional tags used to assist with text searches and metadata
- Log Sub Folder Name
  - If *Log Messages* is enabled this specified an optional sub-folder for the RAW file containing the message's data
- Delimiters
  - The character(s) to use to split the message into its Data Fields

- Relay via UDP Broadcast
  - A flag which tells ACQ to immediately send the data out via UDP Broadcast the instant it is read from the interface.

NMEA Message Definitions

NMEA 0183 is a combined electrical and data specification for communication between marine electronics such as echo sounder, sonars, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments. It has been defined by, and is controlled by, the National Marine Electronics Association. It replaces the earlier NMEA 0180 and NMEA 0182 standards.

The electrical standard that is used is EIA-422, although most hardware with NMEA-0183 outputs are also able to drive a single EIA-232 port. Although the standard calls for isolated inputs and outputs, there are various series of hardware that do not adhere to this requirement.

The NMEA 0183 standard uses a simple ASCII, serial communications protocol that defines how data are transmitted in a "sentence" from one "talker" to multiple "listeners" at a time. Through the use of intermediate expanders, a talker can have a unidirectional conversation with a nearly unlimited number of listeners, and using multiplexers, multiple sensors can talk to a single computer port.

At the application layer, the standard also defines the contents of each sentence (message) type, so that all listeners can parse messages accurately.

The new standard, NMEA 2000, accommodates several *talkers* at a higher baud rate, without using a central hub, or round-robin packet buffering.

⚠ At the time of this writing SCS does not support NMEA 2000.

Inside SCS a NMEA message is a special type of Delimited Message which is always in the NMEA-183 Standard format or equivalent. In contrast to fixed messages, NMEA messages may have variable record length. The order of the data fields is constant, but the number of characters in each field can change. Commas serve to separate all data fields. Also, a Physical Device often sends more than one message, each distinguished by a unique identifier (called a *Sentence Label*). You must create a Message Definition for each message that ACQ will be accepting from the device. Thus, interfacing a Trimble GPS receiver sending $GPGGA and $GPVTG messages would require two NMEA Message Definitions: one for the Trimble $GPGGA message and the other for the Trimble $GPVTG message.

Example

Below is an example of a GPS GGA (essential fix data which provide 3D location and accuracy data) NMEA message. Note the message begins with $GPGGA (the sentence label) which is used to define the following comma delimited data. ALL GGA messages, regardless of device, should output their data in the exact same format. This makes NMEA sensors much easier to understand and integrate into SCS as they all should comply with the standard.

$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Where:

GGA Global Positioning System Fix Data
123519 Fix taken at 12:35:19 UTC
4807.038,N Latitude 48 deg 07.038' N
01131.000,E Longitude 11 deg 31.000' E
1 Fix quality: 0 = invalid
1 = GPS fix (SPS)
2 = DGPS fix
3 = PPS fix
4 = Real Time Kinematic
5 = Float RTK
6 = estimated (dead reckoning) (2.3 feature)
7 = Manual input mode
8 = Simulation mode
08 Number of satellites being tracked
0.9 Horizontal dilution of position
545.4,M Altitude, Meters, above mean sea level
46.9,M Height of geoid (mean sea level) above WGS84 ellipsoid
(empty field) time in seconds since last DGPS update
(empty field) DGPS station ID number
*47 the checksum data, always begins with *

SCS has the majority of known NMEA Message Definitions already pre-defined for you. When creating a new NMEA Message Definition you must select a *Sentence Label* which matches the feed you are trying to ingest. For your convenience SCS will automatically create a full definition matching what it thinks the settings would be to ingest a stream with that sentence label. It also creates all known Data Fields for the message. This results in a large number of Data Fields which are not actually needed by your setup, feel free to select and delete any which are not required in your use cases.

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the *Display Name* attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique.
- Type
  - The Type of Message Definition being defined.
  - This is set upon creation and cannot be changed after the fact.
- Log Messages

    - A flag lettings ACQ know if this stream should be logged to disk (files / database) or not.
    - If unchecked this data will still be available to all clients and calculated devices, but will not be stored on disk.
- Enabled
    - A flag letting SCS know if this interface is actively providing data or if it should be ignored by client applications such as ACQ.
- Termination Characters
    - The last character (often a control character) which signifies the end of the data string.
    - Each NMEA Message Definition should end in a Line Feed (0x10) as per the standard.
- Sentence Label
    - This is the sentence label ACQ will use to filter out this definition from the incoming raw data stream being provided by the sensor.

The *Advanced* tab consists of the following fields:

- Comment
    - A general comment pertaining to this item
- Keywords
    - Optional tags used to assist with text searches and metadata
- Log Sub Folder Name
    - If *Log Messages* is enabled this specified an optional sub-folder for the RAW file containing the message's data
- Delimiters
    - The character(s) to use to split the message into its Data Fields
    - NMEA standard dictates this should be a comma (,)
- Relay via UDP Broadcast
    - A flag which tells ACQ to immediately send the data out via UDP Broadcast the instant it is read from the interface.
- Extended Label
    - A flag which tells SCS that this messages sentence label should be combined with one or more Data Fields to generate a unique sentence label.
    - The NMEA-183 message spec allows sensor manufacturers to compose proprietary message formats that look like NMEA messages but do not conform to the 183 standard in one or more ways. Manufacturers may indicate this by using the characters “$P” at the start of the sentence label. (Note: “P” standing for “proprietary”.)
    - In the NOAA fleet the SIMRAD PI4x net mensuration sensor is in use on several fisheries trawl ships. This message violates the NMEA standard in that its sentence label is not sufficient by itself to describe the information present in the message. The complete description is specified only when you know the value of one or more fields in the message. SCS NMEA messages by default must be completely described by the sentence label. In order to accommodate the scenario presented by the SIMRAD sensor SCS has implemented the Extend Label and Label Extension parameters in the NMEA Message Definition To quote from the SIMRAD manual: All PI3X/4X/5X-specific NMEA sentences start with”$PSIMP”, followed by a 1or 2 character field specifying which PIXX sentence is being sent. The sentence specifies are C: Configuration sentence D: Data sentence (obsolete) D1: Data sentence F: Sensor

definition sentence (obsolete) F1: Sensor definition sentence S: Frequency spectrum sentence T: “Transparent” or system sentences, for communication between DSP and external equipment without interfering with the PI3X/4X/5X display unit. It gets worse: Field one of the sentences indicates the basic sensor type. There is also a “sensor type variant” in field four of the message. These two fields are letters, and together with the sentence label, determine the content of the rest of the fields in the message. EXAMPLE: Field 1 = “D1” indicates this is a data sentence. Field 4 = “H1” indicates the data is “head rope height” while Field 4 = “T1” indicates the data is temperature.

Delimited Definitions

In contrast to delimited and NMEA messages, fixed messages have a set record length. The order of the Data Fields is constant as are their start/end positions.

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the *Display Name* attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique.
- Type
  - The Type of Message Definition being defined.
  - This is set upon creation and cannot be changed after the fact.
- Log Messages
  - A flag lettings ACQ know if this stream should be logged to disk (files / database) or not.
  - If unchecked this data will still be available to all clients and calculated devices, but will not be stored on disk.
- Enabled
  - A flag letting SCS know if this interface is actively providing data or if it should be ignored by client applications such as ACQ.
- Termination Characters
  - The last character (often a control character) which signifies the end of the data string.

- Record Size
  - The max length of the string comprising the entire message.
  - ACQ will assume a message is complete when the *termination character* is read OR the *record size* is exceeded.
  - As most all modern data comes in with a termination character, it is advised you set the *record size* to a very large number thereby telling ACQ to ignore it and always look for the *termination character*.

The *Advanced* tab consists of the following fields:

- Comment
  - A general comment pertaining to this item
- Keywords
  - Optional tags used to assist with text searches and metadata
- Log Sub Folder Name
  - If *Log Messages* is enabled this specified an optional sub-folder for the RAW file containing the message's data
- Delimiters
  - The character(s) to use to split the message into its Data Fields
- Relay via UDP Broadcast
  - A flag which tells ACQ to immediately send the data out via UDP Broadcast the instant it is read from the interface.

Delimited Data Fields

Data Fields represent the smallest element a sensor's data stream can be broken down into. Each Data Field should correspond to a single data item such as a Latitude, Longitude, Course, Speed, Depth, etc. Delimited Data Fields are parsed out of the full RAW data string by looking for one or more delimiters (such as a comma). For instance, given the full RAW message below there are 14 Data Fields

```
MINST,CHATN,CHAT1,CHATW,CHATM,CHATE,003,004,005,006,007,,,
```

Index	Data Field Value
0	`MINST`
1	`CHATN`
2	`CHAT1`
.....	.....

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the *Display Name* attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique.
- Type
  - The Type of DataField Interface being defined.
  - This is set upon creation and cannot be changed after the fact.
- Category
  - The category/genre of data this field should fall into.
  - This must be set appropriately as many calculated interfaces and other sub-systems rely on this setting to filter and generate their inputs/outputs.
- Units
  - The unit of data this field's value is in
- Is Eligible Base Field
  - A flag letting CFE know if this field should be selectable when generating Calculated Interfaces and equations.
  - This flag will also block usage of this field in QA/QC.
- Calculated Interface References
  - Will appear if this field is used by any calculated interfaces, hidden otherwise.
  - Will show you all Calculated Interfaces using this field and provides a button to quickly navigate you to them.
- Data Field Position
  - The index (0-based).

The *Advanced* tab consists of the following fields:

- Comment
  - A general comment pertaining to this item
- Keywords
  - Optional tags used to assist with text searches and metadata
- Precision
  - The number of digits after the decimal point
- Include as Data Source for Category
  - A flag letting SCS know if the field should be considered as a reference source for it's category.
  - Priority of this field is relative to others in the same category based upon the installed Physical Device Reference Order
- Exclude from SAMOS calculations
  - A flag letting SCS know if the field should be averaged and sent to SAMOS.

    - Only applies to fields SAMOS are interested in (eg has no effect on data that wouldn't be going to SAMOS regardless, such as 'winch tension')
- Code Translations
    - A table of data field values and their translations. It is used to translate coded data field values into a string meaningful to users. The translated string replaces the original code in the RAW file and in all data displays

Latitude / Longitude Categories

If you set the fields category to a Latitude or a Longitude you must also specify the format. In the Advanced tab you must also specify the Lat/Lon Direction Field Position (the North/South East/West portion of the data).

Time / DateTime Categories

If you set the fields category to a Time or Date/Time you should specify the format of the timestamp string.

Geographic Point Category

⚠ The Geographic Point Category is used internally by SCS, you should not use it in your normal course of building out your configuration in CFE.

NMEA Data Fields

Data Fields represent the smallest element a sensor's data stream can be broken down into. Each Data Field should correspond to a single data item such as a Latitude, Longitude, Course, Speed, Depth, etc. NMEA Data Fields are parsed out of the full RAW data string by splitting on the comma. For instance, below is the start of defining the indices of the fields from a full GPS RAW message:

⚠ Note index 0 is the sentence label itself!

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
```

Index	Data Field Value
0	`$GPGGA`
1	`123519`
2	`4807.038`
.....	.....

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the *Display Name* attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique.
- Type
  - The Type of DataField Interface being defined.
  - This is set upon creation and cannot be changed after the fact.
- Category
  - The category/genre of data this field should fall into.
  - This must be set appropriately as many calculated interfaces and other sub-systems rely on this setting to filter and generate their inputs/outputs.
- Units
  - The unit of data this field's value is in
- Is Eligible Base Field
  - A flag letting CFE know if this field should be selectable when generating Calculated Interfaces and equations.
  - This flag will also block usage of this field in QA/QC.
- Calculated Interface References
  - Will appear if this field is used by any calculated interfaces, hidden otherwise.
  - Will show you all Calculated Interfaces using this field and provides a button to quickly navigate you to them.
- Data Field Position
  - The index (0-based, but position 0 is the *sentence label* so all Data Fields should START at 1).

The *Advanced* tab consists of the following fields:

- Comment
  - A general comment pertaining to this item
- Keywords
  - Optional tags used to assist with text searches and metadata
- Precision
  - The number of digits after the decimal point
- Include as Data Source for Category
  - A flag letting SCS know if the field should be considered as a reference source for it's category.
  - Priority of this field is relative to others in the same category based upon the installed Physical Device Reference Order
- Exclude from SAMOS calculations

    - A flag letting SCS know if the field should be averaged and sent to SAMOS.
    - Only applies to fields SAMOS are interested in (eg has no effect on data that wouldn't be going to SAMOS regardless, such as 'winch tension')
- Code Translations
    - A table of data field values and their translations. It is used to translate coded data field values into a string meaningful to users. The translated string replaces the original code in the RAW file and in all data displays

Latitude / Longitude Categories

If you set the fields category to a Latitude or a Longitude you must also specify the format.

> Default format for latitude and longitude data in NMEA is Degrees-Minutes. eg *4124.8963, N* which maps to *41d 24.8963' N* or *41d 24' 54" N*

In the Advanced tab you must also specify the Lat/Lon Direction Field Position (the North/South East/West portion of the data).

Time / DateTime Categories

If you set the fields category to a Time or Date/Time you should specify the format of the timestamp string.

> Default format for Time data in NMEA GPS is HHmmss.f. eg *170834* which maps to *17:08:34 Z*

Geographic Point Category

> ⚠ The Geographic Point Category is used internally by SCS, you should not use it in your normal course of building out your configuration in CFE.

Fixed Data Fields

Data Fields represent the smallest element a sensor's data stream can be broken down into. Each Data Field should correspond to a single data item such as a Latitude, Longitude, Course, Speed, Depth, etc. Fixed Data Fields are parsed out of the full RAW data string by setting a start and end position. For instance, below is the start of defining the indices of the fields from an example of a full TSG RAW message:

```
t1= 27.7553, c1= 5.55991, s= 34.7398, sv=1540.216, t2= 27.6226
```

Start	End	Data Field Value	Type of Data
5	11	`27.7553`	Temperature
17	24	`5.55991`	Conductivity
30	36	`34.7398`	Salinity

....

Finding the start and end character indices can be difficult to do by eye. You can copy a line of the output and paste it into a text editor to get them or use a tool such as the Serial Port Diagnostics helper, etc.

⚠ Fixed fields have a hard and locked start/end index but the actual data itself might grow and shrink (imagine a temperature going from 99 degrees to 100 degrees). As such it is highly advisable to have a buffer around your start/end indices (include the empty white space in front and behind your data field if at all possible).

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the *Display Name* attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique.
- Type
  - The Type of DataField Interface being defined.
  - This is set upon creation and cannot be changed after the fact.
- Category
  - The category/genre of data this field should fall into.
  - This must be set appropriately as many calculated interfaces and other sub-systems rely on this setting to filter and generate their inputs/outputs.
- Units
  - The unit of data this field's value is in
- Is Eligible Base Field
  - A flag letting CFE know if this field should be selectable when generating Calculated Interfaces and equations.
  - This flag will also block usage of this field in QA/QC.
- Calculated Interface References
  - Will appear if this field is used by any calculated interfaces, hidden otherwise.
  - Will show you all Calculated Interfaces using this field and provides a button to quickly navigate you to them.
- Starting Position
  - The index where the field starts (recommended to include empty white space in front of the value if at all possible!)
- Ending Position
  - The index where the field ends (recommended to include empty white space behind the value if at all possible!)

The *Advanced* tab consists of the following fields:

- Comment
  - A general comment pertaining to this item
- Keywords
  - Optional tags used to assist with text searches and metadata
- Precision
  - The number of digits after the decimal point
- Include as Data Source for Category
  - A flag letting SCS know if the field should be considered as a reference source for it's category.
  - Priority of this field is relative to others in the same category based upon the installed Physical Device Reference Order
- Exclude from SAMOS calculations
  - A flag letting SCS know if the field should be averaged and sent to SAMOS.
  - Only applies to fields SAMOS are interested in (eg has no effect on data that wouldn't be going to SAMOS regardless, such as 'winch tension')
- Code Translations
  - A table of data field values and their translations. It is used to translate coded data field values into a string meaningful to users. The translated string replaces the original code in the RAW file and in all data displays

Latitude / Longitude Categories

If you set the fields category to a Latitude or a Longitude you must also specify the format. In the Advanced tab you must also specify the Lat/Lon Direction Field Position (the North/South East/West portion of the data).

Time / DateTime Categories

If you set the fields category to a Time or Date/Time you should specify the format of the timestamp string.

Geographic Point Category

> ⚠ The Geographic Point Category is used internally by SCS, you should not use it in your normal course of building out your configuration in CFE.

Calculated / Derived Interfaces

Sensor Interfaces receive messages from external sources through COM or network ports. Calculated Interfaces do not have source external to ACQ.

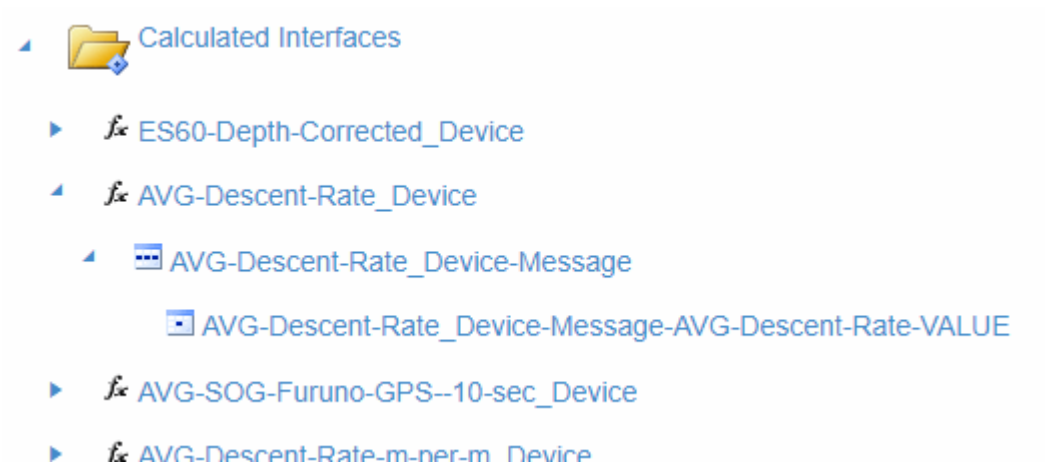
Calculated Interfaces use data from other sensors ("base sensors") and combine them to calculate new data values. A calculated item then constructs a message containing the derived values which appears to SCS to have come from an external physical device. The message can be logged and parsed into data fields as if it were coming from an external source. As such, Calculated Interfaces follow the same structure as normal Sensor Interfaces with an interface level, a message level and one or more field levels.

See also: Sensor Interfaces

There is a special type of calculated interface called SAMOS. These interfaces are automatically managed by the underlying SCS system and should be considered read-only. Any changes you make may be overwritten by the system at any time. Given that, in this version of SCS you no longer have to create nor maintain your SAMOS suite yourself, it is all automatically done for you.

Maintaining your Calculated Interfaces

CFE allows you to configure and maintain the collection of Calculated Interfaces which ACQ then interprets, executes and disseminates the results for other SCS components read, store, analyze and display.

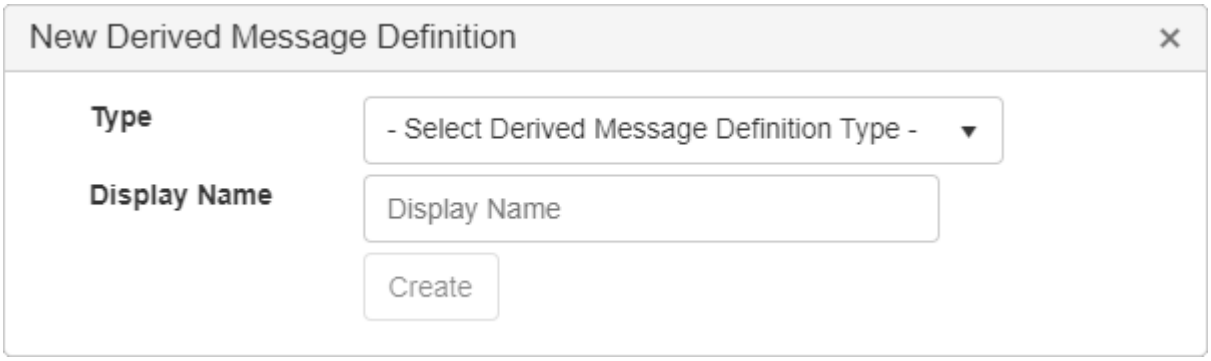


Expanding any Calculated Interface will display the Message Definitions being calculated via that interface. You may only have a single (1) message per interface.

Expanding the Message Definition will display all the Data Fields defined inside that definition.

Creating Interfaces

To add a new Calculated Interface you can click the ***Calculated Interface*** link under the *Add New Item* dropdown in the main toolbar, or simply right click on the *Calculated Interfaces* tree node and choose *Create Calculated Interface*.



As with other items in CFE a base set of information is required before the object can be created.

For Calculated Interfaces you must supply the type and a display name.

Once you have supplied the basics above CFE will automatically add your new item to the appropriate place in the tree and select it for further editing. The detailed pane will now contain a new and mostly empty form. Fill out the remaining information the best that you can, any required fields will be highlighted for you. Don't forget to Save your changes!

Editing

As with other CFE items, a Calculated Interface details pane has two tabs, a *Basic* and an *Advanced*. Depending on which type of interface you are editing you will have a different set of items to edit. Please consult the help page matching the selected interface type for more information.

Removing

To remove a Calculated Interface simply right click on it in the tree list and choose the *Delete* option.

Predefined / Canned Calculated Expressions

When creating a new calculated item SCS allows you to choose from a predefined set of canned functions or create your own custom expression. This choice cannot be changed once created. However, if you choose to create a pre-programmed function you can change the function used (see below) by the message at any time.

There are 22 types of predefined functions:

Average (Over Time Span)	Computes the numeric average of a time series for a given time spa
Average (Over Sample)	Computes the numeric average of a time series for a given number
Polar Average (Over Time Span)	Computes the polar average of a time series, provided in units of d
Polar Average (Over Sample)	Computes the polar average of a time series, provided in units of d
True Wind Speed and Direction	Computes the scalar true wind speed and direction.
Vector Average (Over Time Span)	Computes the vector average of a time series for a given time span
Vector Average (Over Sample)	Computes the vector average of a time series for a given number o
Standard Deviation (Over Time Span)	Computes the standard deviation of a time series for a given time s
Standard Deviation (Over Sample)	Computes the standard deviation of a time series for a given numb
Kurtosis (Over Time Span)	Computes the kurtosis of a time series for a given time span.
Kurtosis (Over Sample)	Computes the kurtosis of a time series for a given number of sampl
Skewness (Over Time Span)	Computes the skewness of a time series for a given time span.
Skewness (Over Sample)	Computes the skewness of a time series for a given number of sam
Pearson Correlation Coefficient (Over Time Span)	Computes the Pearson Correlation Coefficient of a time series for a
Pearson Correlation Coefficient (Over Sample)	Computes the Pearson Correlation Coefficient for a given number o
Min (Over Time Span)	Computes the minimum numeric value of a time series for a given t
Min (Over Sample)	Computes the minimum numeric value of a time series for a given n
Max (Over Time Span)	Computes the maximum numeric value of a time series for a given
Max (Over Sample)	Computes the maximum numeric value of a time series for a given
Sea Level Pressure	Computes the sea level pressure based on a barometer a specified
Heat Index	Computes the heat index based on temperature and relative humic
Linear Equation	Computes the scalar value of the linear equation, m * x + b.

> When ACQ outputs a calculated message it formats it similar to a NMEA string. This means it starts with a sentence label ($DERIV) followed by its comma delimited data with a line feed control character at the end.

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the Display Name attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique.
- Type
  - The Type of Message Definition being defined.
  - This is set upon creation and cannot be changed after the fact.

- Log Messages
  - A flag lettings ACQ know if this stream should be logged to disk (files / database) or not.
  - If unchecked this data will still be available to all clients and calculated devices, but will not be stored on disk.
- Enabled
  - A flag letting SCS know if this interface is actively providing data or if it should be ignored by client applications such as ACQ.
- Sentence Label
  - The value to insert as the first field in the output (read-only)
- Function Selection
  - Allows you to choose which predefined function (see above) you wish to use for this calculation.
- Calculation Parameters
  - Allows you to specify the source data used when executing this calculation.
  - Instead of calculating a result every second like in prior versions of SCS the current version allows you to specify one or more triggers. When data from a source marked as a trigger is obtained the calculation is executed. eg if you have a COG from a normal GPS coming in every second and set that as your source then the calculation will be 1x/second but if you have a COG from a POSMV coming in at 10x/second and set that as your source instead then the calculation will occur 10x/second.
- Function Arguments
  - Additional arguments you need (depending on the selected function type) such as the # of samples or length of the time span.

The *Advanced* tab consists of the following fields:

- Comment
  - A general comment pertaining to this item
- Keywords
  - Optional tags used to assist with text searches and metadata
- Log Folder Path
  - If Log Messages is enabled this specified an optional sub-folder for the RAW file containing the message's data
- Delimiters
  - The character(s) to use to split the message into its Data Fields - should be a comma, do not change this!
- Relay via UDP Broadcast
  - A flag which tells ACQ to immediately send the data out via UDP Broadcast the instant it is read from the interface.

Calculation Parameters and Function Arguments

Each type of function has a set of parameters and arguments which are required to successfully create a valid expression.

⚠ Some source data fields are filtered based upon the expected category. If you are expecting to see something (like your GYRO for a Heading) and it is not in the drop down please double check the source and ensure you have assigned it the correct category.

Calculated data fields can be used in other calculated interfaces allowing for infinite nesting and calculations to be built off of each other.

Average (Over Time Span)

Calculation Parameters	A single numeric value you wish to average
Function Arguments	A time span and it's units used to specify the duration of the calculation (eg average every 60 seconds)

Average (Over Sample)

Calculation Parameters	A single numeric value you wish to average
Function Arguments	A numeric value, every time the base sensor value is updated it's sample size is incremented, when that count is equal to this argument the calculation is executed and the counter is reset (eg average every 100 values)

Polar Average (Over Time Span)

Calculation Parameters	A single numeric value (polar/circular) you wish to average
Function Arguments	A time span and it's units used to specify the duration of the calculation (eg average COG every 60 seconds)

Polar Average (Over Sample)

Calculation Parameters	A single numeric value (polar/circular) you wish to average
Function Arguments	A numeric value, every time the base sensor value is updated it's sample size is incremented, when that count is equal to this argument the calculation is executed and the counter is reset (eg average COG every 100 values)

True Wind Speed and Direction

Calculation Parameters	5 source data fields: Speed, Course, Heading, Relative Wind (Speed and Direction)
Function Arguments	- None -

Vector Average (Over Time Span)

Calculation Parameters	Two numeric values (a scalar/magnitude and a direction) you wish to average
Function Arguments	A time span and it's units used to specify the duration of the calculation (eg average True Wind every 60 seconds)

Vector Average (Over Sample)

Calculation Parameters	Two numeric values (a scalar/magnitude and a direction) you wish to average
Function Arguments	A numeric value, every time the base sensor value is updated it's sample size is incremented, when that count is equal to this argument the calculation is executed and the counter is reset (eg average True Wind every 100 values)

Standard Deviation (Over Time Span)

Calculation Parameters	Two numeric values (a source series and an average) from which you want the standard deviation (σ)
Function Arguments	A time span and it's units used to specify the duration of the calculation

> To compute this you may be required to create another `Calculated Interface` which generates the average which is then used as a parameter in this interface.

Standard Deviation (Over Sample)

Calculation Parameters	Two numeric values (a source series and an average) from which you want the standard deviation (σ)
Function Arguments	A numeric value, every time the base sensor value is updated it's sample size is incremented, when that count is equal to this argument the calculation is executed and the counter is reset

To compute this you may be required to create another Calculated Interface which generates the average which is then used as a parameter in this interface.

Kurtosis (Over Time Span)

Calculation Parameters	Three numeric values (a source series, an average and a standard deviation) from which you want the kurtosis
Function Arguments	A time span and it's units used to specify the duration of the calculation

To compute this you may be required to create other Calculated Interfaces which generate the average and standard deviation sources which are then used as parameters in this interface.

Kurtosis (Over Sample)

Calculation Parameters	Three numeric values (a source series, an average and a standard deviation) from which you want the kurtosis
Function Arguments	A numeric value, every time the base sensor value is updated it's sample size is incremented, when that count is equal to this argument the calculation is executed and the counter is reset

To compute this you may be required to create other Calculated Interfaces which generate the average and standard deviation sources which are then used as parameters in this interface.

Skewness (Over Time Span)

Calculation Parameters	Three numeric values (a source series, an average and a standard deviation) from which you want the skewness
Function Arguments	A time span and it's units used to specify the duration of the calculation

To compute this you may be required to create other Calculated Interfaces which generate the average and standard deviation sources which are then used as parameters in this interface.

Skewness (Over Sample)

Calculation Parameters	Three numeric values (a source series, an average and a standard deviation) from which you want the skewness

Function Arguments	A numeric value, every time the base sensor value is updated it's sample size is incremented, when that count is equal to this argument the calculation is executed and the counter is reset

> To compute this you may be required to create other Calculated Interfaces which generate the average and standard deviation sources which are then used as parameters in this interface.

Pearson Correlation Coefficient (Over Time Span)

Calculation Parameters	Four numeric values (two source series and their averages) from which you want the bivariate correlation
Function Arguments	A time span and it's units used to specify the duration of the calculation

> To compute this you may be required to create other Calculated Interfaces which generate the averages which are then used as parameters in this interface.

Pearson Correlation Coefficient (Over Sample)

Calculation Parameters	Four numeric values (two source series and their averages) from which you want the bivariate correlation
Function Arguments	A numeric value, every time the base sensor value is updated it's sample size is incremented, when that count is equal to this argument the calculation is executed and the counter is reset

> To compute this you may be required to create other Calculated Interfaces which generate the averages which are then used as parameters in this interface.

Min (Over Time Span)

Calculation Parameters	One numeric value (two source series and their averages) to compute the minimum value
Function Arguments	A time span and it's units used to specify the duration of the calculation

Min (Over Sample)

Calculation Parameters	One numeric value (a source series, an average and a standard deviation)) to compute the minimum value

Function Arguments	A numeric value, every time the base sensor value is updated it's sample size is incremented, when that count is equal to this argument the calculation is executed and the counter is reset

Max (Over Time Span)

Calculation Parameters	One numeric value (two source series and their averages) to compute the maximum value
Function Arguments	A time span and it's units used to specify the duration of the calculation

Max (Over Sample)

Calculation Parameters	One numeric value (a source series, an average and a standard deviation) to compute the maximum value
Function Arguments	A numeric value, every time the base sensor value is updated it's sample size is incremented, when that count is equal to this argument the calculation is executed and the counter is reset

Sea Level Pressure

Calculation Parameters	Two source data fields: Air Pressure, Temperature (in Celsius, Fahrenheit, Rankine, or Kelvin)
Function Arguments	Height of barometer above sea level (in meters).

Heat Index

Calculation Parameters	Two source data fields: Relative Humidity, Temperature (in Celsius, Fahrenheit, Rankine, or Kelvin)
Function Arguments	- None -

Linear Equation

Calculation Parameters	One numeric value, the variable x of the linear equation, m * x + b
Function Arguments	Real numbers: Slope m, Offset b

Custom / User-Defined Calculated Expressions

When creating a new calculated item SCS allows you to choose from a predefined set of canned functions or create your own custom expression. This choice cannot be changed once created. If you choose to create a custom expression then you can generate a very simple or extremely complex function to take whatever source data you want, manipulate it and output your desired results.

When ACQ outputs a calculated message it formats it similar to a NMEA string. This means it starts with a sentence label ($DERIV) followed by its comma delimited data with a line feed control character at the end.

It is recommended you use the Pre-Programmed functions if they meet your requirement instead of defining your own.

⚠ Integration with R and other external tooling has not been included in this release of SCS. You can only use the SCS expression builder and the limitations therein.

The *Basic* tab consists of the following fields:

- Description
  - Provide an optional description of this item.
- System Name
  - System generated name, read only format standard across all SCS systems.
  - Useful for post-processing data on shore from multiple different sources/vessels.
  - Useful for rotating personnel as each ship will always comply with this format.
  - If you wish to have a more user friendly name you can edit the Display Name attribute below
- Display Name
  - User supplied name
  - Can be anything you wish though must be unique.
- Type
  - The Type of Message Definition being defined.
  - This is set upon creation and cannot be changed after the fact.
- Log Messages
  - A flag lettings ACQ know if this stream should be logged to disk (files / database) or not.
  - If unchecked this data will still be available to all clients and calculated devices, but will not be stored on disk.
- Enabled
  - A flag letting SCS know if this interface is actively providing data or if it should be ignored by client applications such as ACQ.

- Termination Characters
  - The last character (often a control character) which signifies the end of the data string.
- Sentence Label
  - The value to insert as the first field in the output (read-only)
- Calculation Parameters
  - Allows you to specify the source data used when executing this calculation.
  - Instead of calculating a result every second like in prior versions of SCS the current version allows you to specify one or more triggers. When data from a source marked as a trigger is obtained the calculation is executed. eg if you have a COG from a normal GPS coming in every second and set that as your source then the calculation will be 1x/second but if you have a COG from a POSMV coming in at 10x/second and set that as your source instead then the calculation will occur 10x/second.
  - In custom definitions you can provide a 'friendly name' for each source field, this is the name of the variable you want to associate with this field when building out your custom equation.
- Custom Equation
  - The equation you want to execute when generating the output for this calculation (see below).

The *Advanced* tab consists of the following fields:

- Comment
  - A general comment pertaining to this item
- Keywords
  - Optional tags used to assist with text searches and metadata
- Log Folder Path
  - If Log Messages is enabled this specified an optional sub-folder for the RAW file containing the message's data
- Delimiters
  - The character(s) to use to split the message into its Data Fields - should be a comma, do not change this!
- Relay via UDP Broadcast
  - A flag which tells ACQ to immediately send the data out via UDP Broadcast the instant it is read from the interface.

Custom Equation Editor

The *Custom Equation* editor is where you supply the expression you wish to execute to take the source data and build your output data.

It is recommended you:

1. Determine the formula you wish to execute in order to be aware of the source data you will need.
2. Build *Calculation Parameters* for each source field and assign it a unique and descriptive *Friendly Name*

3. Start typing out your formula into the equation editor combining appropriate mathematical functions and the variables you defined in step 2.
4. Tailor your result Data Fields

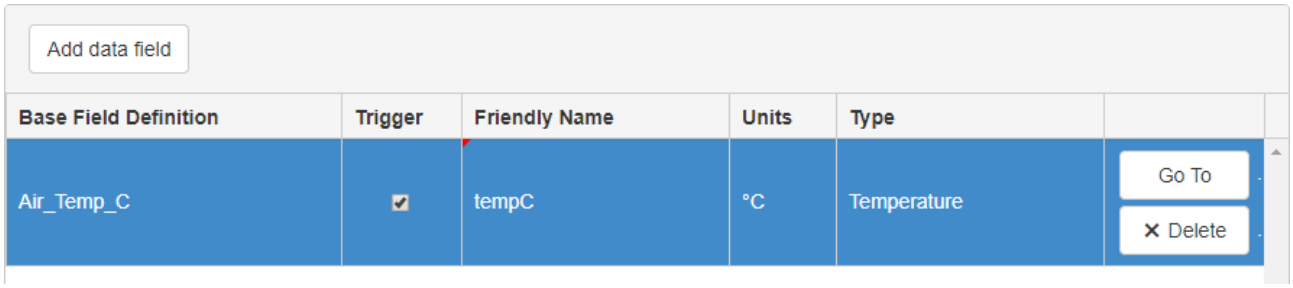
For example, to convert a temperature from Celsius to Fahrenheit:

Determine the formula:

(0°C × 9/5) + 32 = 32°F

Build parameters:

Only need a single source data item, a data field with a temperature reading in units of Celsius.

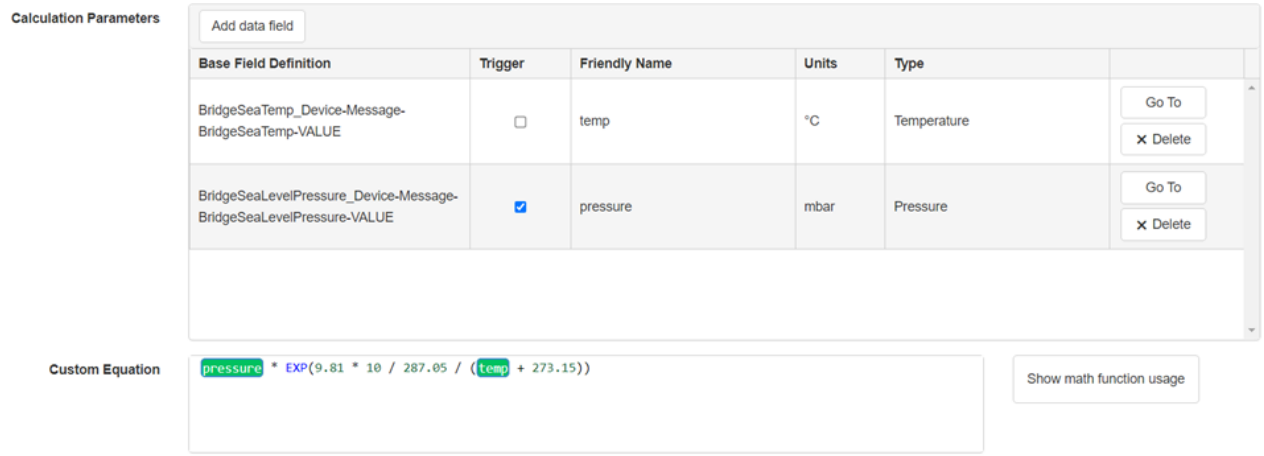


Build Formula

The custom equation editor allows the user to enter an algebraic expression out of user-defined variable names and system-defined constants and mathematical functions. This pseudo-code is compiled by the ACQ Service to be a resultant derived data field value. *The expression is entirely what would appear on the right side of the equals (=) sign. The expression must contain only a single line.* The algebraic expressions allow for addition (+), subtraction (-), multiplication (*) and division (/), and follow the rules for order of operations. The user-defined variables names, or friendly names, are forced to be lowercase. The system-defined constants and functions are all uppercase. Given the friendly name variables and the provided functions and constants, the expression should otherwise follow the syntax the C# programming language.

As you start typing the editor will highlight and attempt to assist you with your equation. If you start typing the *friendly name* of one of your source fields it will display the list for you to choose from and highlight them in green. When you end a scope the parenthesis will be highlight in yellow showing you whats inside. Math functions will also be displayed for you to choose from, many are the standard set you would find in Microsoft Excel.

As another example, consider the following custom equation that calculates the air pressure at sea level, given a sensor at 10 m above sea level. The expression, which is already a pre-programmed function available in CFE, is only to illustrate that mathematical operators.

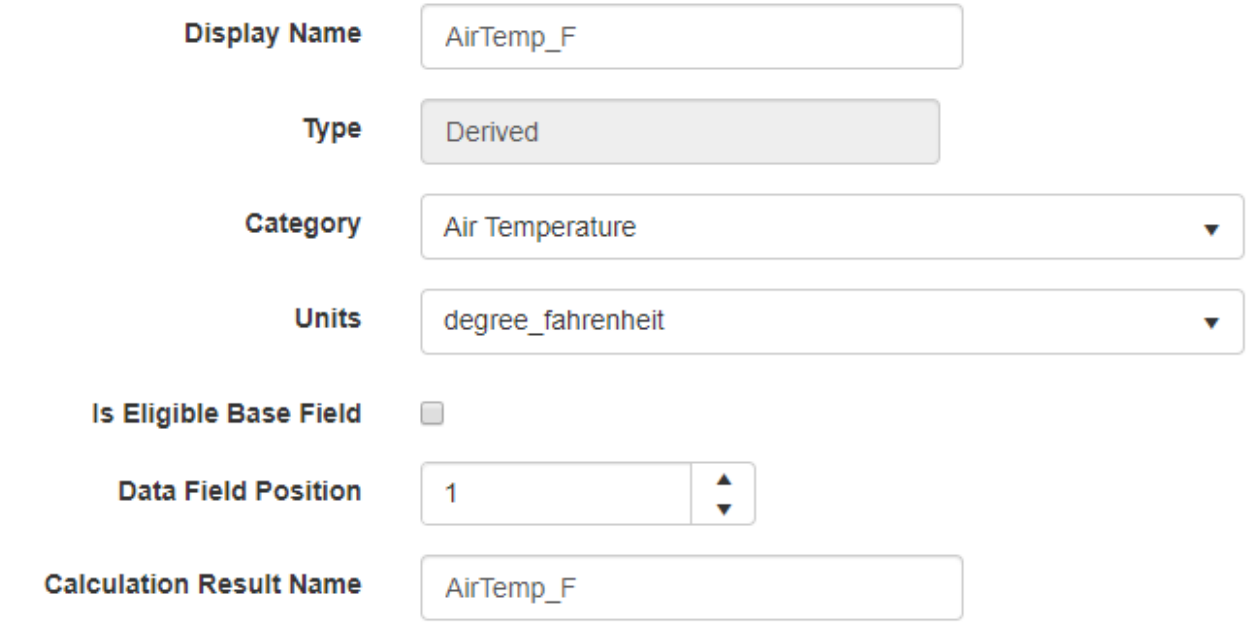


Tailor Output Data Fields

Once your equation is complete proceed to the automatically created Data Fields (the outputs) for this interface.

Name appropriately and be sure to select the correct category and units!

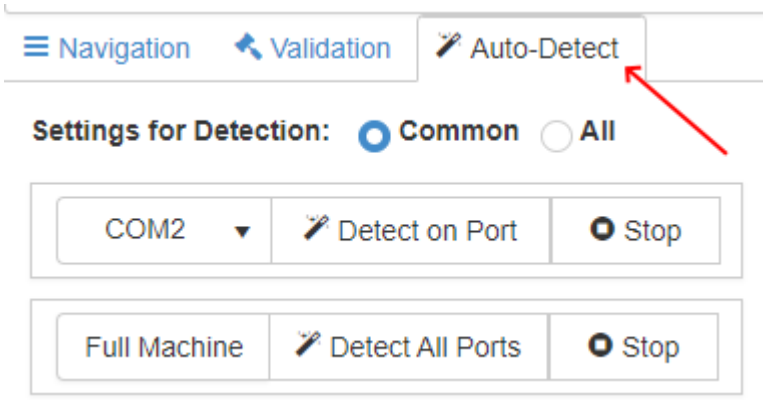
```
1  ( tempC * (9/5) ) + 32
```



Auto Detection of New Sensor Data

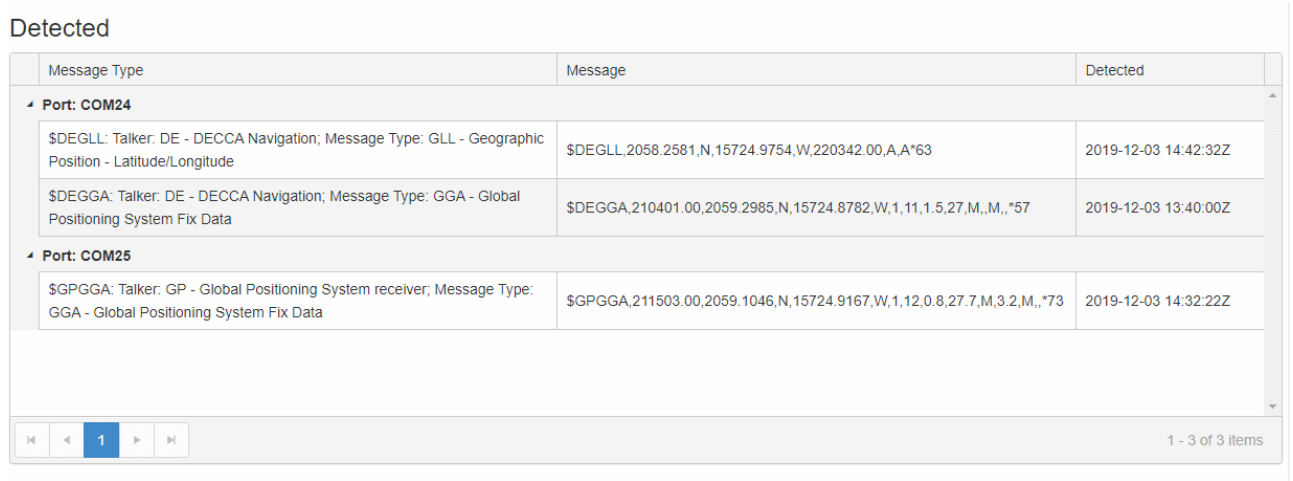
SCS now has the capability to automatically detect new sensor feeds on COM ports hooked up to the primary server. This is accomplished via the Sensor Automatic Detection service. This service runs on-demand and can be run with all available settings (every possible combination of baud, parity, etc) or just the most commonly used ones (much faster though less thorough) for a COM port. If it finds something

new it attempts to match it up against known definitions inside SCS. It will then present the find to you in the *Auto-Detect* tab of CFE.



In addition to the service scanning available ports, ACQ also will send unrecognized data to the service on the ports that are currently active.

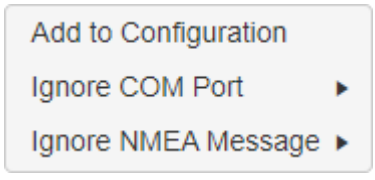
Ongoing results from this service are displayed in the details pane once the tab is selected. At the top of the pane you will see all discovered messages for each port.



If nothing is in the *Detected* grid then no new items have been discovered. Check back periodically to ensure nothing is queued up and waiting for your attention.

Each discovered message is grouped into the port on which it was found. The first column presents what SCS thinks the message is, a sample of the data is displayed in the second column and finally the time that it was detected is in the last column.

Right-clicking on any item will present you with a context menu with action items you can take.



If you wish to add the discovered message definition to your current Working Configuration in CFE click *Add to Configuration*. The entire message and all it's Data Fields will automatically be added and broken out for you. Review it and don't forget to Save and publish for it to become active inside ACQ!

Many devices output data that may not be considered important from a scientific data collection perspective (like the waypoint messages from a GPS). Most likely all these streams will be automatically discovered and added to this list. If you wish to ignore the entire COM port or the message itself you can do so for a period of time (if you are testing or playing with the device/sensor/port, etc) or indefinitely (never want to see it again, like perhaps the above mentioned waypoint data).

If you choose to ignore the items, whether temporarily or permanently, they will be added to the *Ignored* grid located below the *Detected* grid.

Right-clicking on an ignored item will let you stop ignoring it and move it back into the *Detected* grid.

Auto detection is only available over COM ports. Polled COM, network and any other means of incoming data traffic are ignored.

Auto detection has to iterate all possible combinations of serial settings and therefore can take quite some time to process a port before starting over from the beginning. If have plugged something in and it's not showing up you may wish to kick off a scanning job specific to the data you know is there. To do this you can tell the background service to immediately prioritize and start an auto-detection job on a specific port via the *Detect On Port* button inside the *Auto-Detect* tab. Simply pick the port from the drop-down list and click the button.

The Ignored grid shows ports that are ignored for all reasons, not just those specifically set to be ignored by a user. The 3 types are “Ignored by User” “Locked by ACQ” and “Locked by Custom Message.” Hovering over a row will bring up details e.g. "In use by COM Interface HullTempC-High." Ports that are locked by ACQ or locked by a custom message are at a lower opacity and do not provide a context menu since they are for informational purposes only. Ports that are Ignored by User are listed first, are at full opacity, and provide a context menu to allow the user to stop ignoring the port.

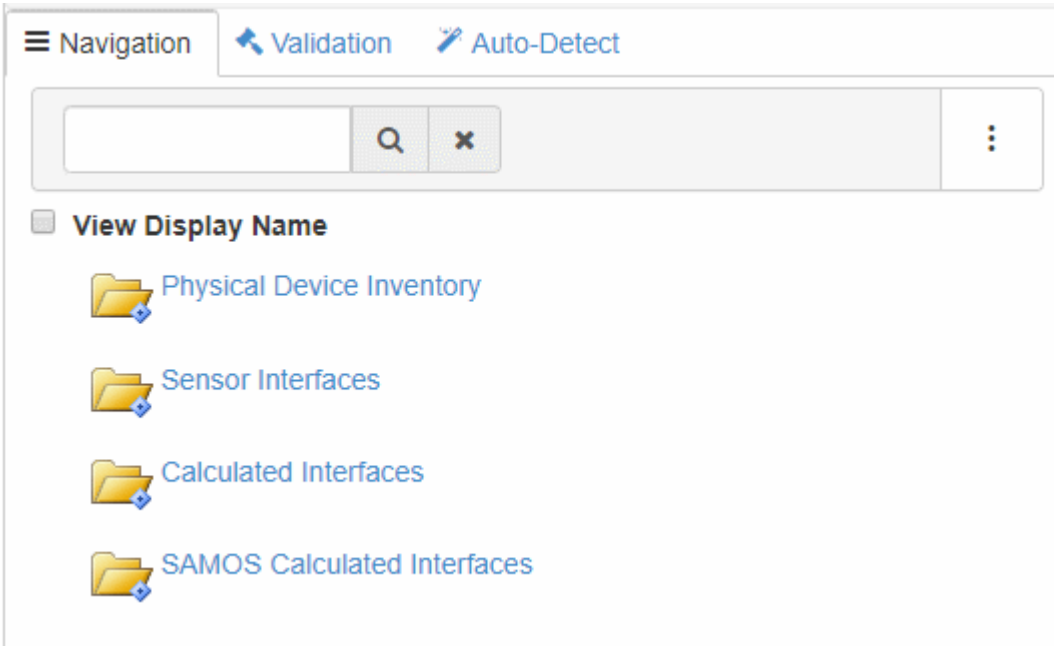
⚠ The service cannot scan ports currently in use by ACQ. If you have the interface settings correct (baud, parity, etc) then ACQ itself should relay undefined messages to the service. But if you have incorrect settings then ACQ will hold the port and no data will be sent to the service. If you want to do a full scan of a port you have to remove it from being active inside ACQ (disable or delete the interface and publish).

⚠ The service cannot scan ports currently in use by Custom Messages. If you want to do a full scan of a port being used in a Custom Message you have to stop the message before attempting to run the scan.

⚠ This service might have a tendency to consume CPU. If you are running on a more modest machine or notice high CPU usage you might want to consider disabling this service on startup.

Clearing

There are times and use cases where you would wish to reset your configuration to a blank slate. To clear your sensor configuration simply click ***Clear*** from the *Configuration* dropdown located in the main toolbar.



Clearing your configuration will remove all items, including your physical inventory and all calculated interfaces.

Once a *Clear* operation has been performed it is immediately followed by a *Save* and your new Working Configuration will effectively be empty.

Note this does NOT trigger a *Publish* and SCS will continue to operate under the fully loaded / prior configuration until you do so.

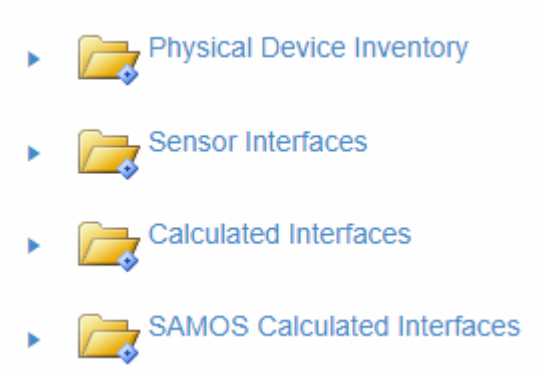
Clearing your configuration allows you to start from scratch and is a prerequisite for some other operations you may wish to perform like importing a Zip Ship or Loading a Previous configuration.

> If you clear your configuration by mistake you can always revert to the currently active configuration via a Load Previous operation. This will restore your Working Configuration to the Published Configuration (the one currently being used by SCS). Changes you might have made to the current Published Configuration, if any, will have been lost however.

Editing

As described in the Introduction, there are 3 main user interface elements inside CFE. The bulk of your work will be done via the tree found in the *Navigation* tab and the and element editing display pane.

The tree has multiple root nodes, each dedicated to a specific portion of your sensor configuration. Expanding any node will expose the sub-components covered by that nodes area of focus. CFE also is one of the few portions of SCS where a context menu is implemented, right clicking on nodes and in various other locations yields many useful commands related to the element under the mouse.



The Physical Device node is where you manage the metadata covering all actual physical sensors you have on board your ship.

The Sensor Interfaces node is where you manage how and what SCS itself ingests from the above noted sensors.

The Calculated Interfaces node is where you create custom equations to post-process the incoming data into other useful streams for your mission.

The SAMOS node is a special implementation of Calculated Interfaces specifically for submission to SAMOS. For all intents and purposes it should be treated as read-only.

Via these various nodes you are able to add, remove and edit the vast majority of things covered by CFE. Once your configuration is completed and deployed chances are it will change very little over time. If you get a new sensor on board you would like to hook up, this would be the place to add it. If a sensor breaks and is no longer on board, this would be the place to remove it. Generally speaking these operations don't occur on a daily basis, once your system is setup you may not touch CFE again for a decent amount of time. However, since CFE is so critical and impactful to the efficient and accurate operation of the entire SCS suite it is advised you check in and refresh your situational awareness and competency in this tool frequently so when the time comes that you need to execute something with it you are ready to do so.

When editing elements inside CFE tree your main workflow would be to select (or create) an item in the tree, this will cause the detail pane on the right to load with it's information. You can edit the form in the details pane to be correct, then optionally save your changes and/or publish them for use.

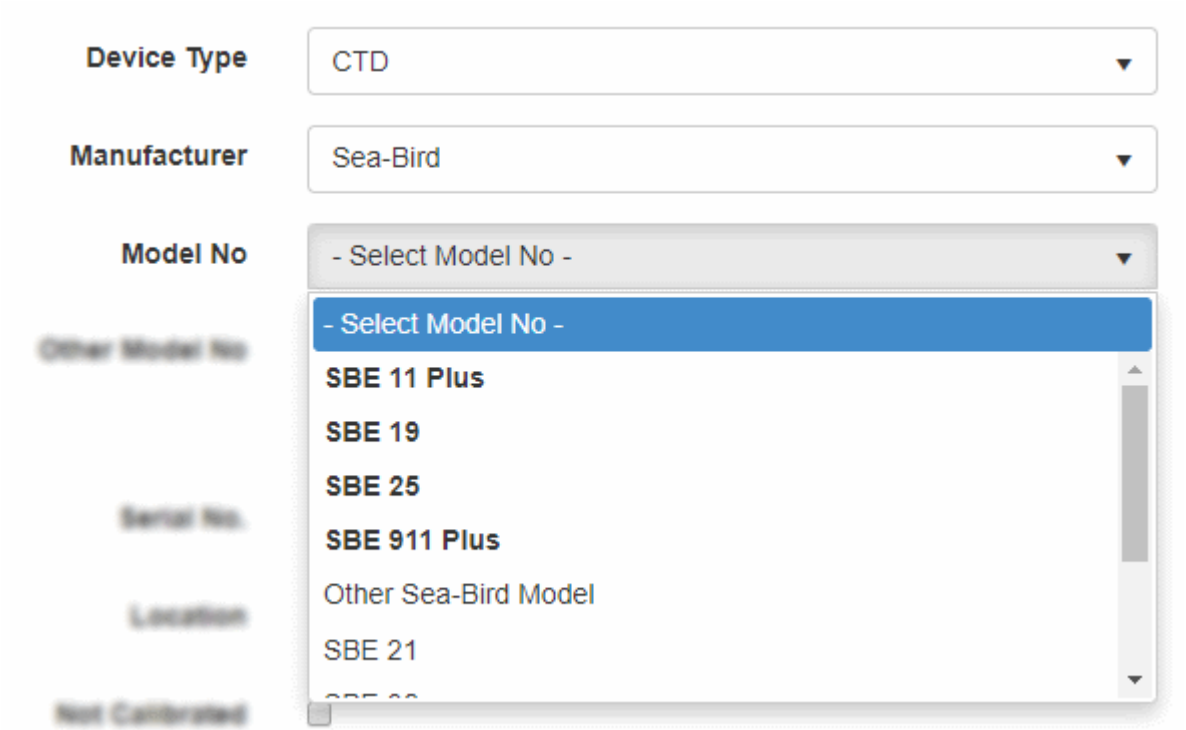
Details Pane

The details pane usually breaks down into two tabs: *Basic* and ***Advanced.*** This was done to hide away things which are rarely used so as to make the interface more intuitive and less distracting.

- The *Basic* tab contains the most commonly edited attributes for the selected item
- The *Advanced* tab holds the less frequently used attributes. Though less commonly changed it is a good idea to review these attributes for accuracy as well.

Recommended Selections

Some attributes in the details pane will have drop down lists. Some common ones are attributes like units, manufacturers and models.



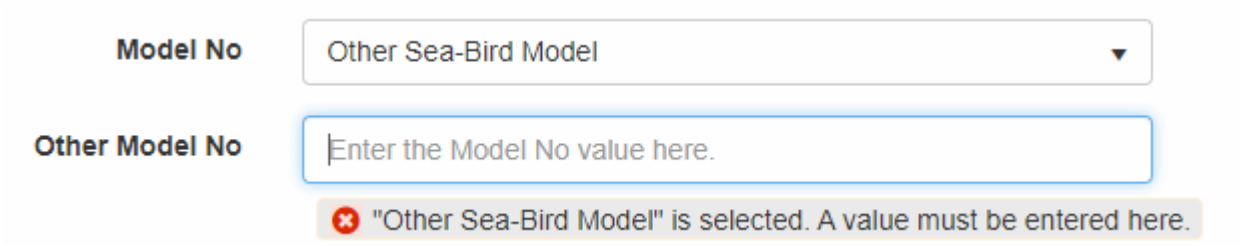
These lists will attempt to organize themselves in a manner which makes your most likely choices bold and at the top.

For instance, if you have a CTD device by Sea-Bird then SCS can guess the most likely model numbers that you would be working with. It will automatically move those to the top of the list and highlight them for your selection.

This is of course an educated guess. In the event the top items are not correct you can choose any other item in the drop down list including a special item named ***Other***

Other Selections

There are many places in CFE where SCS presents a list of options to you to choose from. For obvious reasons many of these lists are subject to change and while they may seem complete at the time of this writing they may fall out of sync with reality over time. For this reason, and many others, SCS allows you as an administrator to manually code in your own values for some of these lists. To do this you must select the *Other* option.

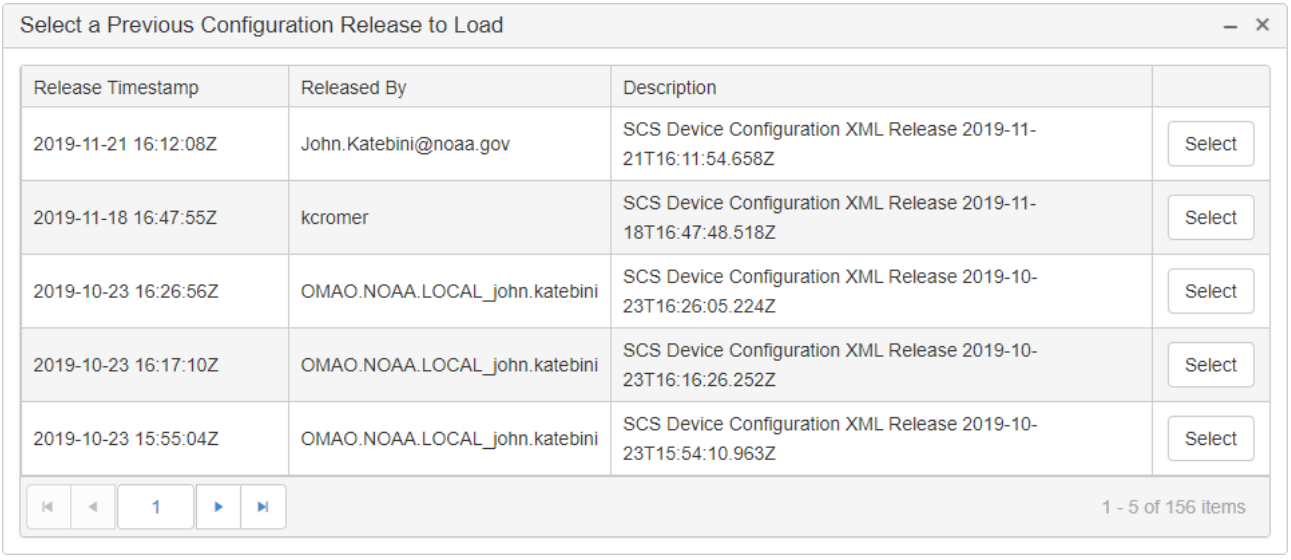


When you choose *Other* you must supply a free-text value which you feel is the correct one for the item being entered. This value will later be evaluated by a shore-side quorum and either added to the master list and sent out to all SCS users as a new option or rejected and automatically changed.

This is done in an attempt to keep the SCS lists up to date via user feedback (crowd-sourcing of sorts) but also to prevent the user-base at large from creating lists of non-standard vocabularies (eg to prevent some folks using °C and others using degC and others using Celsius when all mean the same thing).

Loading a Previous Configuration

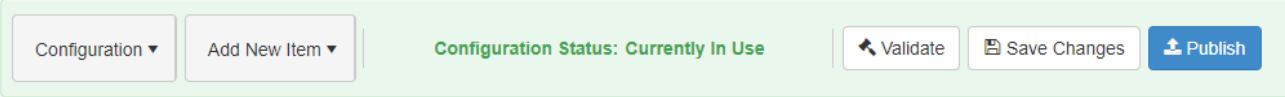
Every time you Publish your Working Configuration it is tagged and saved in the database. Your entire Publish history is recorded and available to you should you wish to use it. To load a previously published configuration you simply click *Load Previous* from the *Configuration* dropdown located in the main toolbar. This will present you with a list of all your prior publishes and allow you to load any of them into your Working Configuration.



⚠ This operation does not perform a *Publish*, if you want the historical publish to be the currently active one be sure to publish it after it has been loaded.

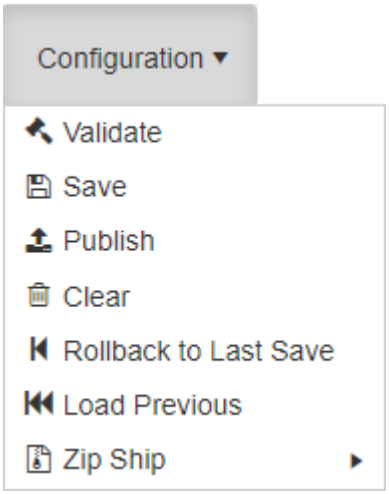
⚠ Loading a previous publish requires your Working Configuration to be cleared. If you have changes be sure to Publish them for use or to create a ZipShip backup or they will be lost.

CFE Toolbar



At the top of the CFE interface there is a toolbar which provides you with many commands used in maintaining your sensor tree and the global state of your configuration.

The *Configuration* drop down is used to control the state of the configuration setup from a global perspective. It consists of multiple commands which act on the sensor configuration as a whole.



The *Validate* command reviews the sensor configuration you have setup at the moment and tries to find any issues with it.

The *Save* command take the entire configuration and saves it to the database. While this saves your changes it DOES NOT make them active!

The *Publish* command takes your working configuration and makes it active.

The *Clear* command wipes your current configuration out allowing you to start fresh or utilize a previous or external configuration.

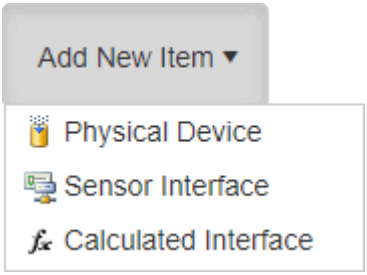
The *Rollback* command wipes out any changes you have made sine your last save and restores the configuration to the last saved state.

The *Load Previous* command lets you load a previously published configuration.

The *Zip Ship - Create* command takes your entire sensor configuration (including images) and serializes it into a zip file which you can use as a backup or to transfer to other vessels.

The *Zip Ship - Import* command takes a previously exported configuration (file created by a Zip Ship - Create command) and loads it into your setup.

The *Add New Item* drop down adds new items to the configuration tree located inside the tab strip below the toolbar. You can add a new physical device, new sensor interface or a new calculated interface to your working configuration.



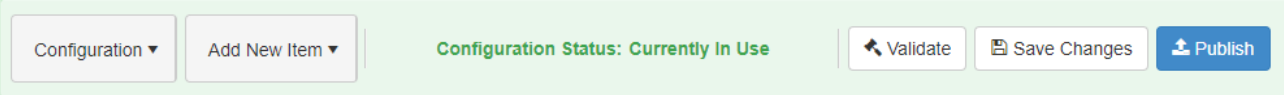
The Validate, Save Changes and Publish buttons are shortcuts to the same operations detailed above in the Configuration dropdown. Since they are so frequently performed explicit buttons were added to make it easier to execute them.

Publishing

The concept of publishing a configuration goes back multiple versions of SCS. As before, SCS administrators can continuously edit the sensor configuration in CFE without impacting the running SCS system (the Working Configuration). However, at some point the changes are ready to go live and be put to use by the rest of SCS. In order to push those changes into production the administrator has to perform a *Publish* operation.

> Publishing your configuration takes any changes you have made since your last publish and makes them part of the active configuration used by all SCS systems. In other words, it tells all of SCS to start using the changes you have made in CFE.

You can only perform a publish operation if you have pending changes. You can determine this by looking at the main toolbar, if it is green then your Working configuration and your Published Configuration match. If not, then you have pending changes you can push to production when you are ready. To publish your sensor configuration simply click ***Publish*** from the *Configuration* dropdown located in the main toolbar or click the ***Publish*** button directly.



> ⚠ Unlike prior versions of SCS, new configurations published by CFE take effect immediately. SCS will not wait for a restart to integrate the changes, do not publish unless you are ready for your changes to go live.

When you perform a Publish operation the first thing that happens is your configuration is saved, then a full validation scan is done. If any validation issues are found you will have a chance to review (or ignore) them prior to completing the publish.

Once a Publish operation completes successfully the toolbar will turn green and all running SCS clients, from ACQ to charts, will take your changes into account. For instance, if you added a new COM port device, ACQ will immediately start reading from it, the datahub will immediately start sending it out and the auto-detection service will immediately stop listening on it. Similarly, if you edit any item which effects a template the template will be alerted that it may be in an invalid state and will let any user know that it may need attention. This doesn't necessarily mean it will stop working, templates will continue to operate regardless of changes made in CFE, however it may impact the result (for instance if you have a Real Time Display showing a sensor value, and you delete the sensor in CFE then the next time you launch the Real Time Display it cannot display that sensor value anymore, but it will start regardless).

> Restarting of templates is no longer required for CFE changes to be reflected in them and templates will start even if publishes from CFE 'break' them.

> If you are running the FSDB service then your new configuration will also be sent to shore and sync'd to any other SCS backup servers you have running on the ship.

When publishing you will have the option of providing a description. Entering something descriptive and meaningful can prove useful in the future when you look through your publish history to find the one you want during a Load Previous operation.

Rollback to Last Save

Hopefully as you edit your Working Configuration you have periodically saved your state. In the event you make changes to your configuration and decide they were a mistake or you want to clear them you can perform a *Rollback* operation. To rollback your sensor configuration simply click ***Rollback to Last Save*** from the *Configuration* dropdown located in the main toolbar. Rolling back to your last save will undo any changes you have made since the last *Save* operation.

> It is advisable to *Save* prior to any major modifications to your configuration. This will allow you to *Rollback* if you are not happy with the changes you make.

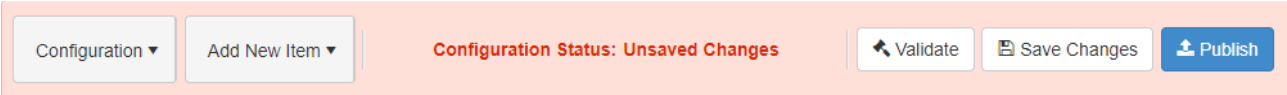
> ⚠ Many operations (clear, publish, etc) automatically perform a *Save*, be aware of this as it may prevent you from rolling back to the place where you manually saved.

> ⚠ There is only 1 saved checkpoint. Every time a *Save* operation is performed that becomes the new saved state. Prior saved states are not kept and you can only rollback to the state of the very last *Save* performed.

Saving

A typical ship's sensor configuration can get quite complicated and large. As when editing things in a word processor, saving often is a good idea.

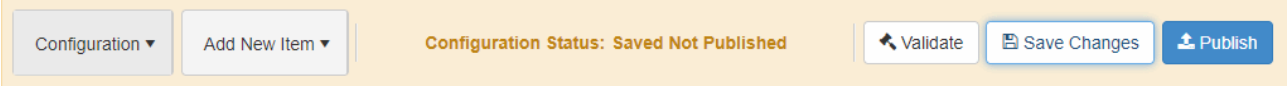
When you start making changes to your configuration you will notice the main toolbar turn to a reddish color and display text alerting you to the fact you have unsaved edits.



If you go to lunch, sleep, etc or even close the browser your changes will not be lost. They are kept in memory on the server itself. The next time you load CFE you will see the same screen with the same warning. However, in the event the server itself is rebooted (patches, etc) then any unsaved edits you have made are at risk.

To save your sensor configuration simply click ***Save*** from the *Configuration* dropdown located in the main toolbar or click the *Save Changes* button directly. Once you have saved your changes they are serialized to persistent storage and will survive even the server being rebooted. This also sets a hard state you can roll back to if down the road, after many other edits, you wish to abandon them and revert back.

Once you have saved your changes the main toolbar will update itself to a yellowish color and the text will reflect your new state. Your changes are now saved to the database, but they are not yet published and actively being used by clients.



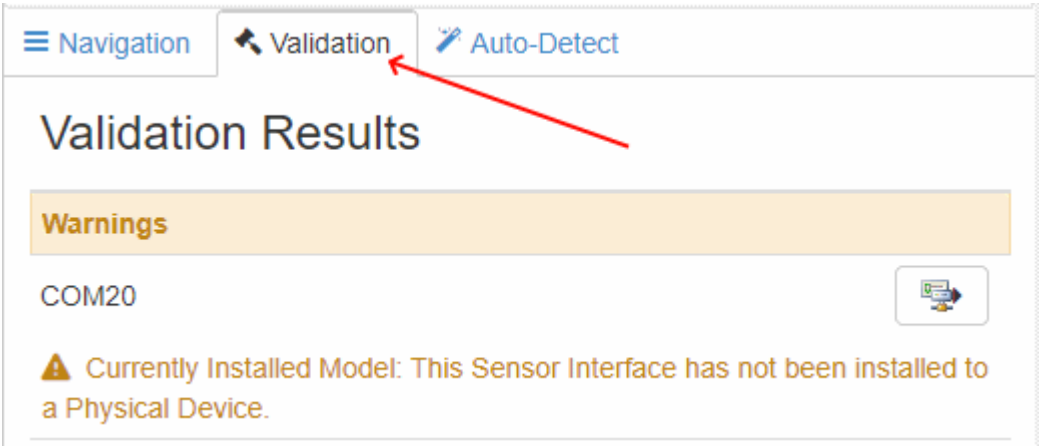
Any changes you make are completely independent of the running SCS system (ACQ, Events, clients, etc). You can edit and save repeatedly for as long as you wish without effecting anything but CFE.

Once you are happy with your changes you can make them live by publishing them; or if you wish you can abandon them by clearing the config and loading the last publish.

Validation

There are a lot of settings available to you when building and maintaining your sensor configuration. With so many options there is a high probability of errors or issues arising when modifications are made to the configuration. In order to minimize impacts on operations the SCS CFE tool has, similar to other templates, the ability to validate your setup prior to saving and making your changes live.

You can run validation against your setup at any time, even if you you are in the middle of building your setup. It is passive and has no impact on your work, it only provides feedback to you. To run validation simply click *Validate* from the *Configuration* dropdown or the *Validate* button located in the main toolbar. The system will work through your configuration and generate a set of potential issues which it will then display to you in the *Validate Tab*.



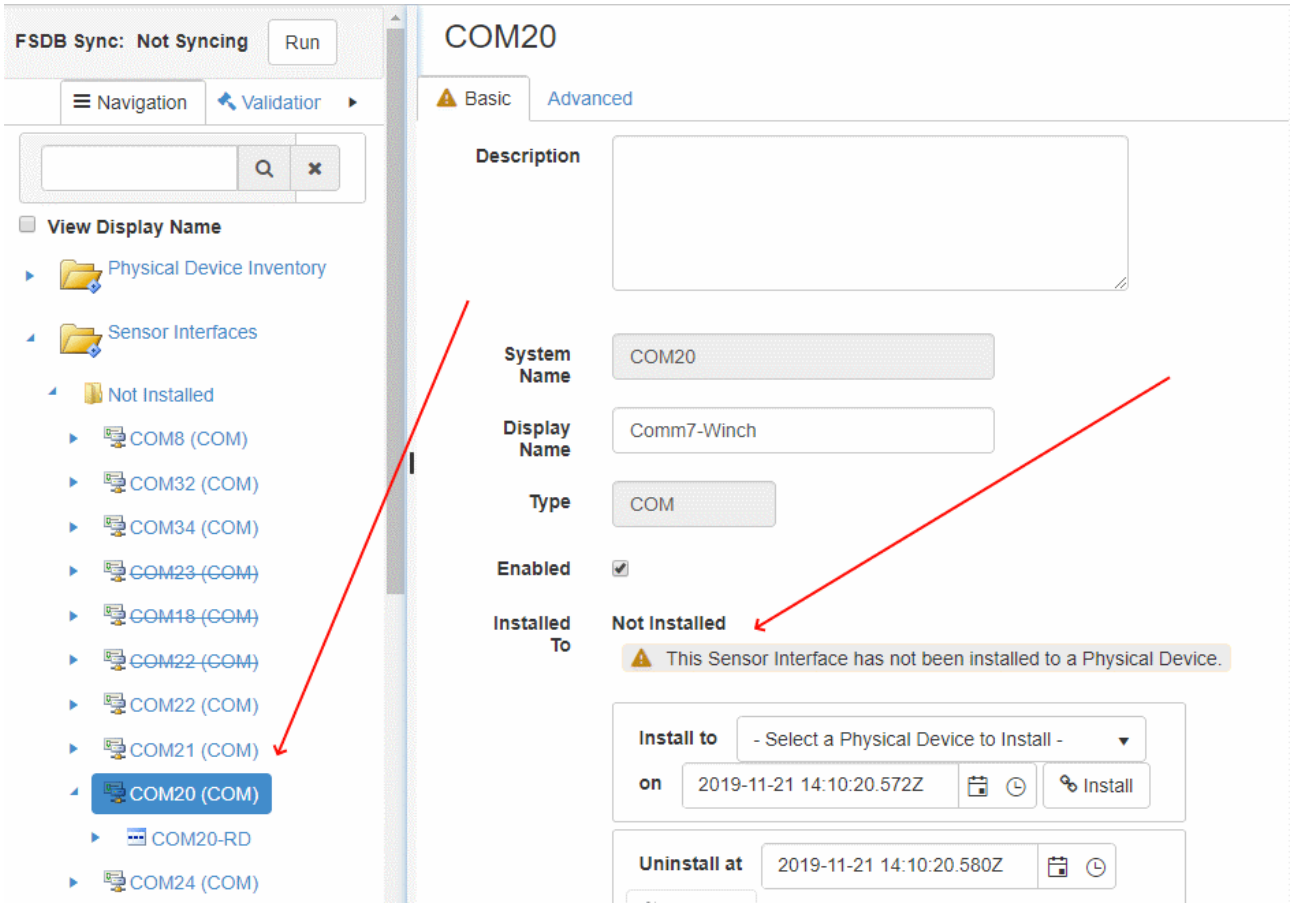
There are 3 levels of issues that may be presented:

Critical	Critical validation issues put the configuration itself into a state in which it cannot be serialized and saved. These must be fixed before you can even save your config much less publish and use.
Error	Error validation issues allow the configuration to be saved, however issues have been detected which would prevent the configuration to be published and used.
Warning	Warnings are issues which don't prevent saving, publishing or usage of your configuration. They are simply items of note which the system thinks might need your attention.

⚠ While ideally all validation issues would be addressed and resolved the Critical and Error findings must be corrected before your configuration can be set as active and used by all clients and sub-systems in SCS.

For each entry on the Validation Tab you will see a brief message which tells you which portion of the config is causing the error and why. You may also notice a small button you may click on as a shortcut to take you to the offending element.

For the above issue, clicking the button or finding COM20 in the tree list will load the interface with the issue. CFE will also display a warning right on the screen pinpointing the item for you making it easier to find and resolve the warning.



Once you have fixed any/all the validation messages feel free to re-validate and see if your setup is cleared or not. Once all Critical and Error items have been fixed you can publish your configuration for active use.

> Publishing a configuration automatically validates it first and displays any issues found asking for your confirmation before actually publishing.

Zip Ship

As you create and publish sensor configurations you may want to save them outside and independent of the SCS system. Performing a Zip Ship operation allows you to do this by taking the entire configuration, images and all, and serializing it into a single zip file. You can then take that file and store it somewhere else, like your ships backup NAS or a CD or even on shore. In the event you ever want to restore that configuration, such as complete hardware failure, etc, all you have to do is perform a Zip Ship Import operation on that file to pull it back in.

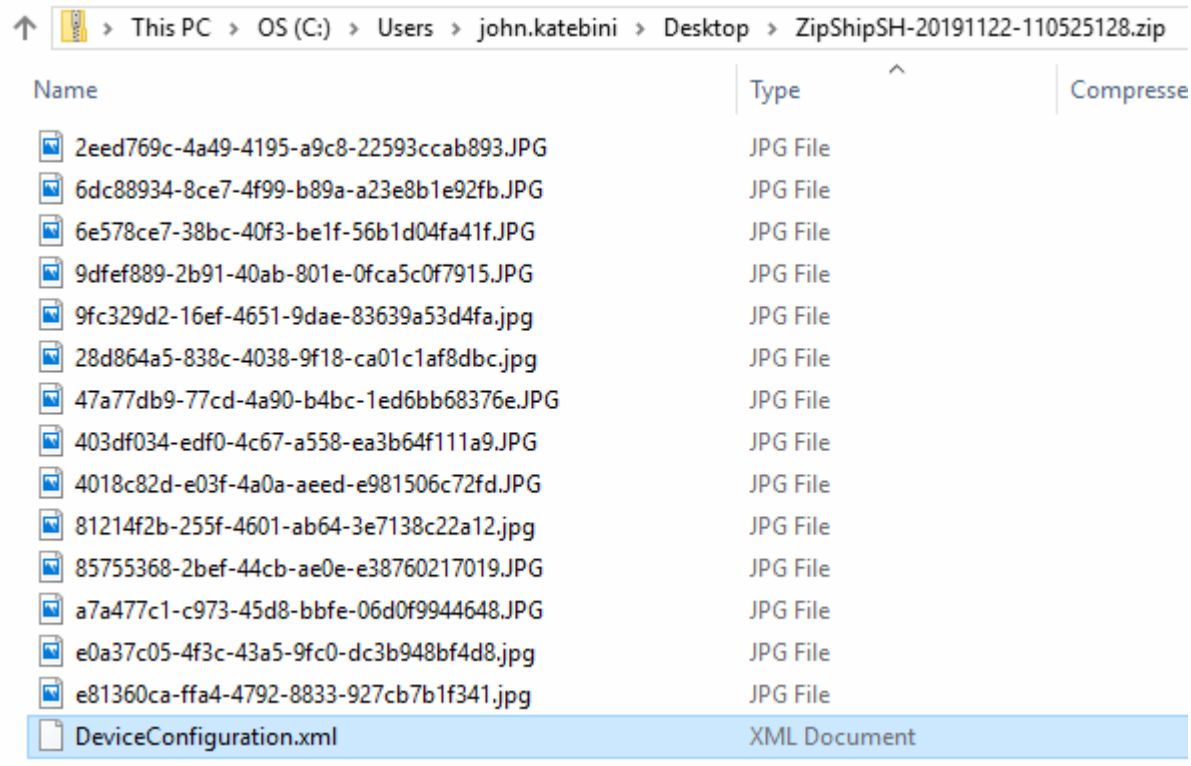
To save your configuration to a file you simply click *Zip Ship => Create* from the *Configuration* dropdown located in the main toolbar. This will automatically create and download a file named ZipShip[%Ship Initials%]-[%Timestamp%].zip

To import a previously exported ZipShip simply click Zip Ship => Import from the *Configuration* dropdown located in the main toolbar and select the file you wish to import.

> Creating a zip ship file serializes the current Published Configuration. If you have changes in your working configuration you must revert them or publish them prior to creating a zipship.

> Importing a ZipShip requires your working configuration to be cleared

The contents of a ZipShip are simply the entire configuration serialized into a single XML file packaged along with every related file and image associated with the config. Obviously the more images and files you have uploaded the larger the ZipShip will be.



If you are running FSDB this same information is what is synchronized between ship and shore. However, only incremental changes are transferred to optimize package sizes and avoid over consumption of bandwidth.

A ZipShip operation is a great way to send your configuration to shore-based support. They can import your configuration and attempt to replicate your situation using their own servers rather then attempting to diagnose or debug issues over a satellite and can also do so w/out impacting your production environment and on-going operations.

Layout Management

Layouts are the main data visualization page for users and sit at the highest level of the user defined interface. They are essentially the data visualization dashboards of SCS and are at their core simply wrappers around Widget Groups. Much like Events and other high level components of the website, Layouts take up a full screen when running.

> Widget Groups are used to wrap 1 or more Widgets together in a set. Layouts are used to wrap 1 or more Widget Groups together in a set.

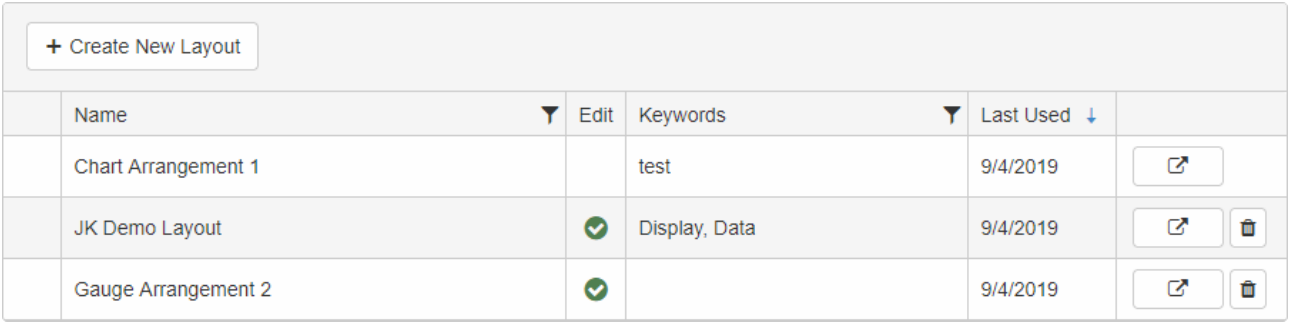
Layouts are similar to other templates in that they have names, permissions and so forth, however they do not have a separate UI dedicated for running vs editing as with widgets. Instead both running and editing can be done from the same UI. There are a variety of ways to manage Layouts. The general user will see a list of the most recently used Layouts (that their account has permission to see) right on the home page in the form of a link. Clicking the link will bring the user to the run-time page and automatically load the Layout.

If the Layouts of interest isn't listed there, or more advanced management is needed (creation, deletion, etc) then the user has to go to the *Manage My Layouts* UI. That UI is accessed via the *Template* link on the main menu or from the *Home Page* via the ***Layouts*** button (*Recently Used Layouts* block).

Creating Layouts

The only way to create a new layout is via the ***Manage My Layouts*** UI.

1. Browse to the UI via the *Template* link on the main menu or from the *Home Page* via the ***Layouts*** button (*Recently Used Layouts* block).
2. Click the *+Create New Layout* button at the top of the grid.



3. You will be directed to the *Layout Editor* which will be loaded with a new blank layout template ready for you to add Widget Groups to.

Editing Layouts

Editing Layouts is done via the *Layout Editor.* This is the main UI / dashboard that most users will interact with.

1. Browse to the UI via the *Template* link on the main menu or from the *Home Page* via the ***Layouts*** button (*Recently Used Layouts* block).

2. Find the Layout you want to change in the grid and click the ***Open/Load*** button at the end of the row.
3. You will see the *Layout Editor* loaded with the Layout you want to edit. Expand the toolbar slider (cog icon in the upper left) and click *Edit*
4. Make your changes and don't forget to hit *Save*!

⚠ Changes made to Layouts will potentially effect all running instances of it throughout the ship.

Edit Mode

The *Layout Editor* is normally in a *Run* mode, however when you wish to build or change a Layout you can switch into *Edit* mode. This is done by expanding the toolbar and clicking *Edit*. When in Edit mode you will be presented with a layout dialog, all Widget Groups will become editable and everything becomes re-arrangeabe.

⚠ Editing and changes in general are restricted based upon your accounts permission level w/regards to the Layout, Widget Groups inside the Layout and even the Widgets themselves on an item by item basis.

Toolbar

When viewing the *Layout Editor* in the upper left corner you will notice a little box with a cog in it. Clicking on the cog will expand a toolbar which provides additional functionality to the Layout currently running or being edited. Clicking the cog again will collapse the slider back and out of the way.

Full Screen
: Put the browser into full screen mode

Run
: Put the editor into "Run" mode, this hooks all the widgets up to their data feeds and makes the layout go 'live'.

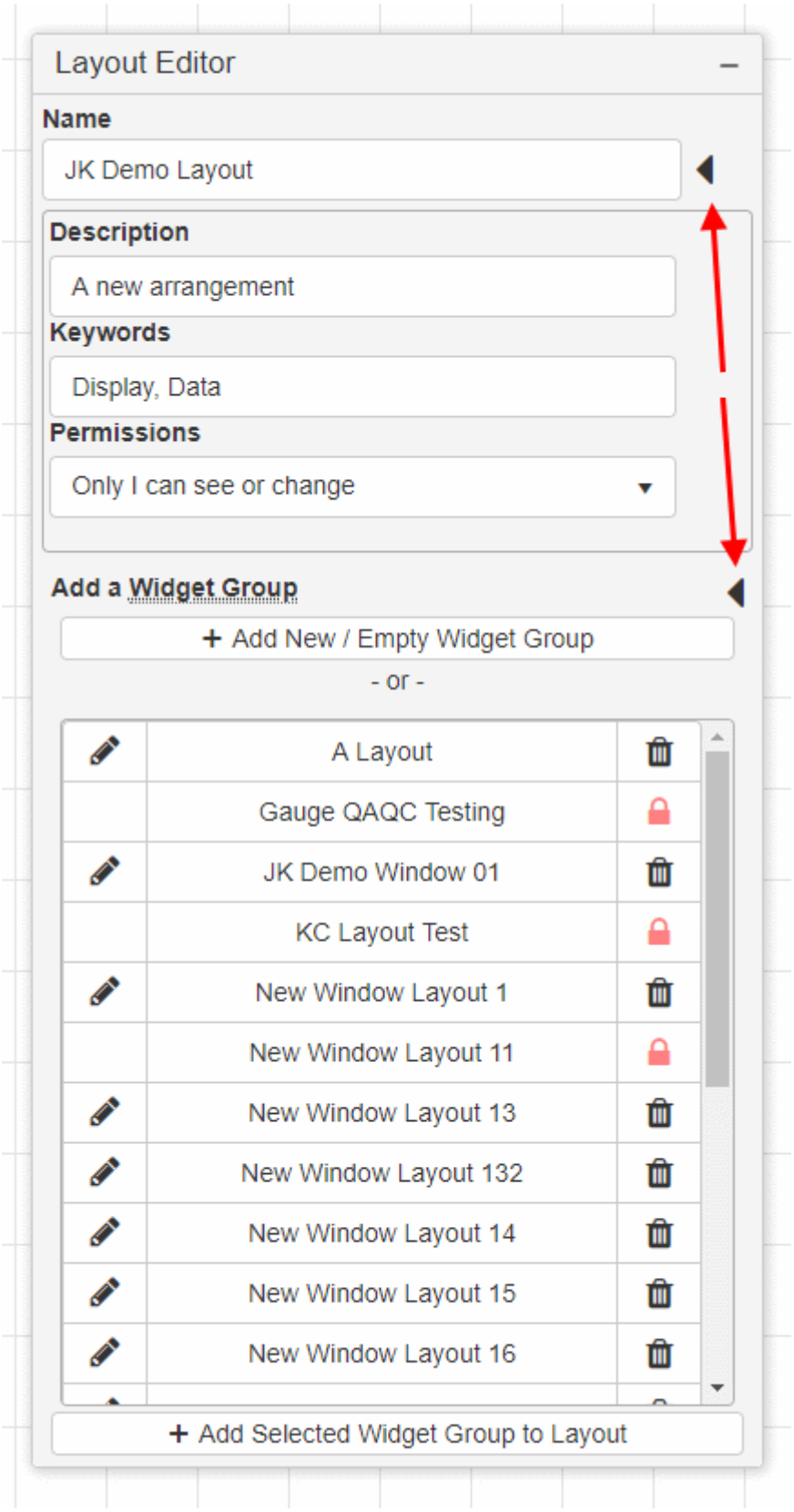
Edit
: Put the editor into "Edit" mode, this allows you to change the look and feel, positions, build or edit widget groups, change permissions, etc

Save
: Save the changes (if any) you have made to the Layout while in Edit mode

Save As
: Clicking the arrow next to *Save* will allow you to save your changes under a different name, this is useful if you don't want to (or can't) effect the original template you were editing

Clear
: Clear all Widget Groups from the Layout and start fresh

Layout Editor Dialog



Immediately below the toolbar, when in Edit mode, a dialog window will appear which allows you to edit the Layout's name, permissions and other meta data associated with templates. In addition it provides a means for you to create new or add/edit/ delete existing Widget Groups to your Layout.

Note the triangular buttons located at the end of the properties group and the widget group (black triangles being pointed to by the red arrows in the image to the left). These buttons will collapse / expand their corresponding section. By default the template properties section is collapsed, if you want to change the Layout's permissions for instance you would have to click the black triangle to see it first.

If you want to create a new Widget Group you can click this button, a new empty group will be placed onto the Layout for you to interact with.

If you want to interact with an existing group first find it in this list. Then there are 3 options you can proceed with:

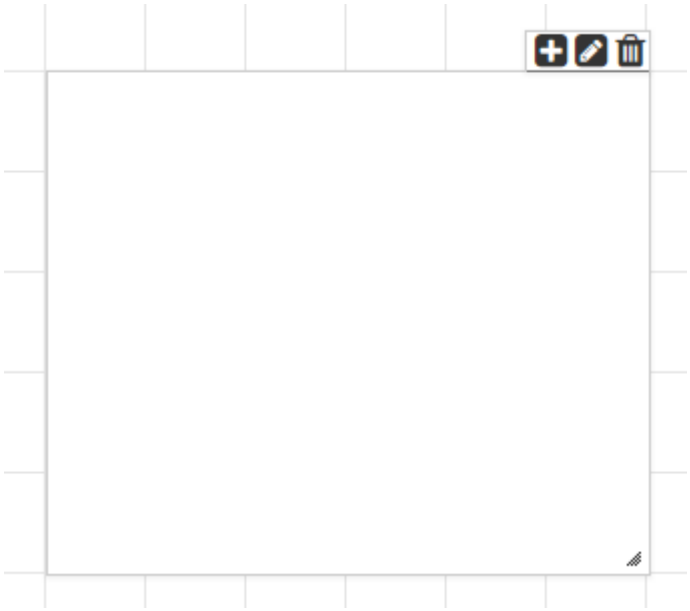
Click the pencil icon to edit the group.
Click the trash can icon to delete the group (permanent delete from database).

- Click the row and then the button at the bottom of the list (*+Add Selected Widget Group*) to add the selected group to your Layout.

If an Edit or Delete icon is not available it is most likely because the account you are currently logged into SCS with does not have appropriate permissions (eg read only) for the given template.

Editing a Widget Group

When a Widget Group is added to your Layout you will see a bounding box containing 0 or more widgets.



The group will have a small toolbar of its own in the upper right corner.

Click the + icon to add new widgets to the group

Click the pencil icon to edit the properties of the group (permissions, name, etc)

Click the trash can icon to remove the group from the Layout

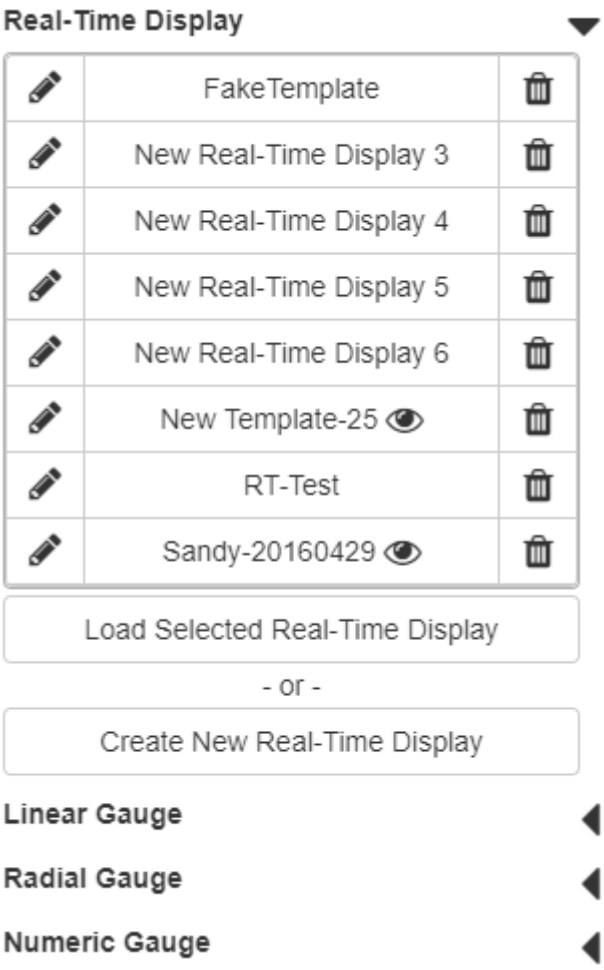
Click anywhere in the white-space inside a group (be careful not to click inside the boundaries of a widget!) and drag it to move it around on the screen.

At the bottom right corner you will notice a small grab handle. Left click this and drag to resize the Widget Group to best fit the widgets you want to have inside of it.

⚠ Clicking the trash can / delete icon removes it from the layout but does NOT remove it from the system. That is accomplished via the *Layout Editor Dialog* above or via the Manage My Widgets UI

Adding Widgets to a Widget Group

While in Edit mode for a Layout, click the + icon above any Widget Group to add a Widget to it. You will be presented with a UI similar to this one.



You will see a list of all widget types on the left with a triangular button in the middle. As with the editor dialog you have to click the triangle to expand the details for the given section.

For example, in the image on the left you can see the Real-Time Display widgets are being displayed after the black triangle (now pointing down) was clicked.

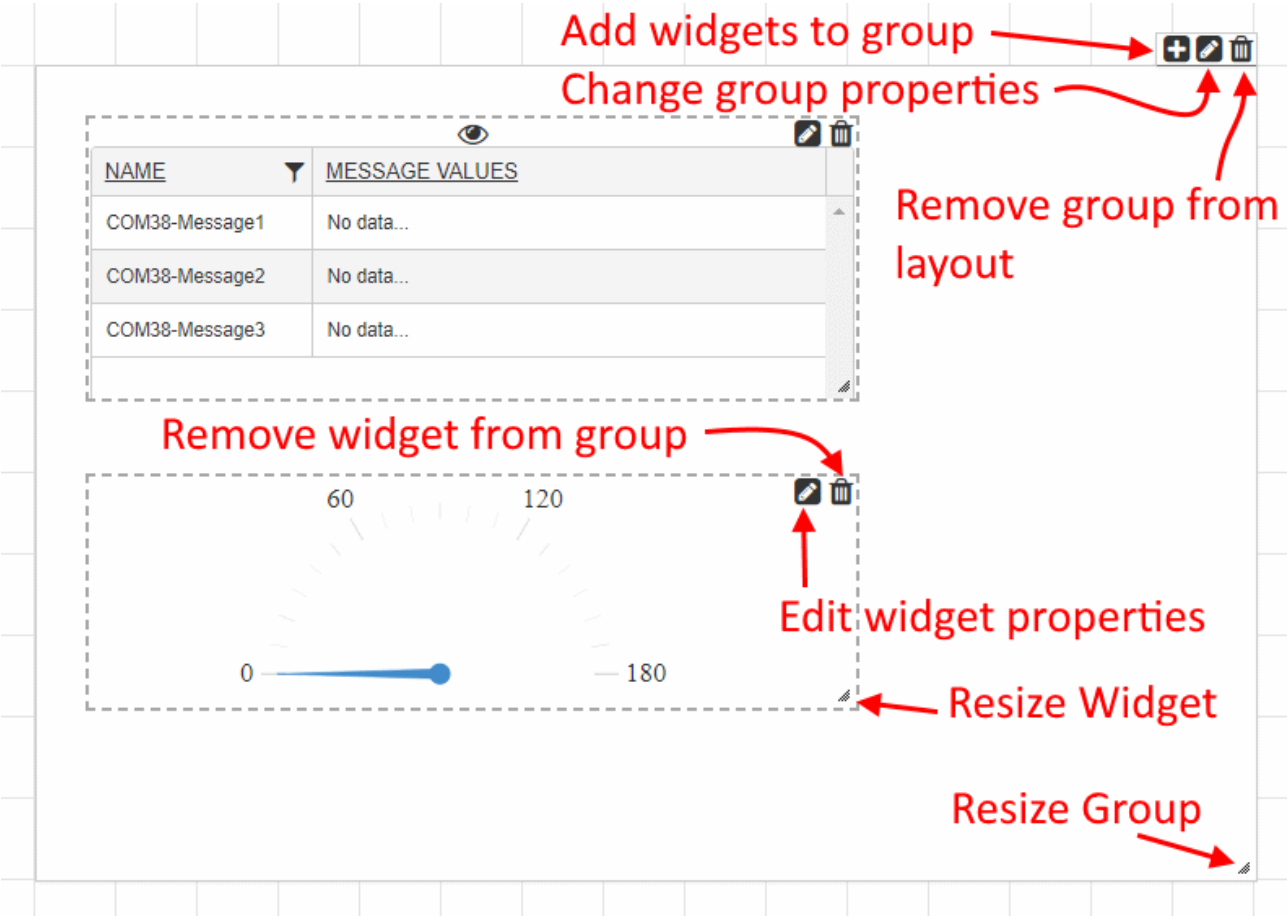
To edit a widget click the pencil icon on its row.

To delete a widget (from database, permanent) click the trashcan icon on its row.

To add the widget to your group select it (click the row) and click the *Load Selected* button at the bottom of the list.

You can also create new widgets of the given type via this dialog. To do so click the Create New button and enter the relevant properties (name, permission, etc). Then hit OK.

Once a widget is added it will have it's own bounding box inside the group with it's own resize handle in the lower right corner. You can also click the widget and drag it around on the screen to move it around. However, just as the Widget Group is constrained by the bounds of the Layout (entire window) the Widgets are constrained by the bounds of their containing Widget Group.



You can add as many or as few Widget Groups to your Layout as you like. You can create as many Layouts as you like. When you are happy with your Layout you can save it via the toolbar for future reuse and/or switch into Run mode via the toolbar to turn it on and go like.

> ⚠ If you simply switch to run mode and do not save you will have to recreate it later if browse away from it. No one else will be able to use it (only exists on your browser) until it is saved at least once.

Deleting Layouts

You can only delete an existing Layout via the ***Manage My Layouts*** UI.

1. Browse to the UI via the *Template* link on the main menu or from the *Home Page* via the ***Layouts*** button (*Recently Used Layouts* block).
2. Find the Layout you want to delete in the grid and click the ***Delete*** button (the one with the trash can).

Widget Management

Widgets are the main data visualization tool for users and sit at the lowest level of the user defined interface. There are a variety of different widget types you can use to develop a dashboard of your data. Certain widget types can be placed together in Widget Groups. Widget Groups are used to wrap 1 or

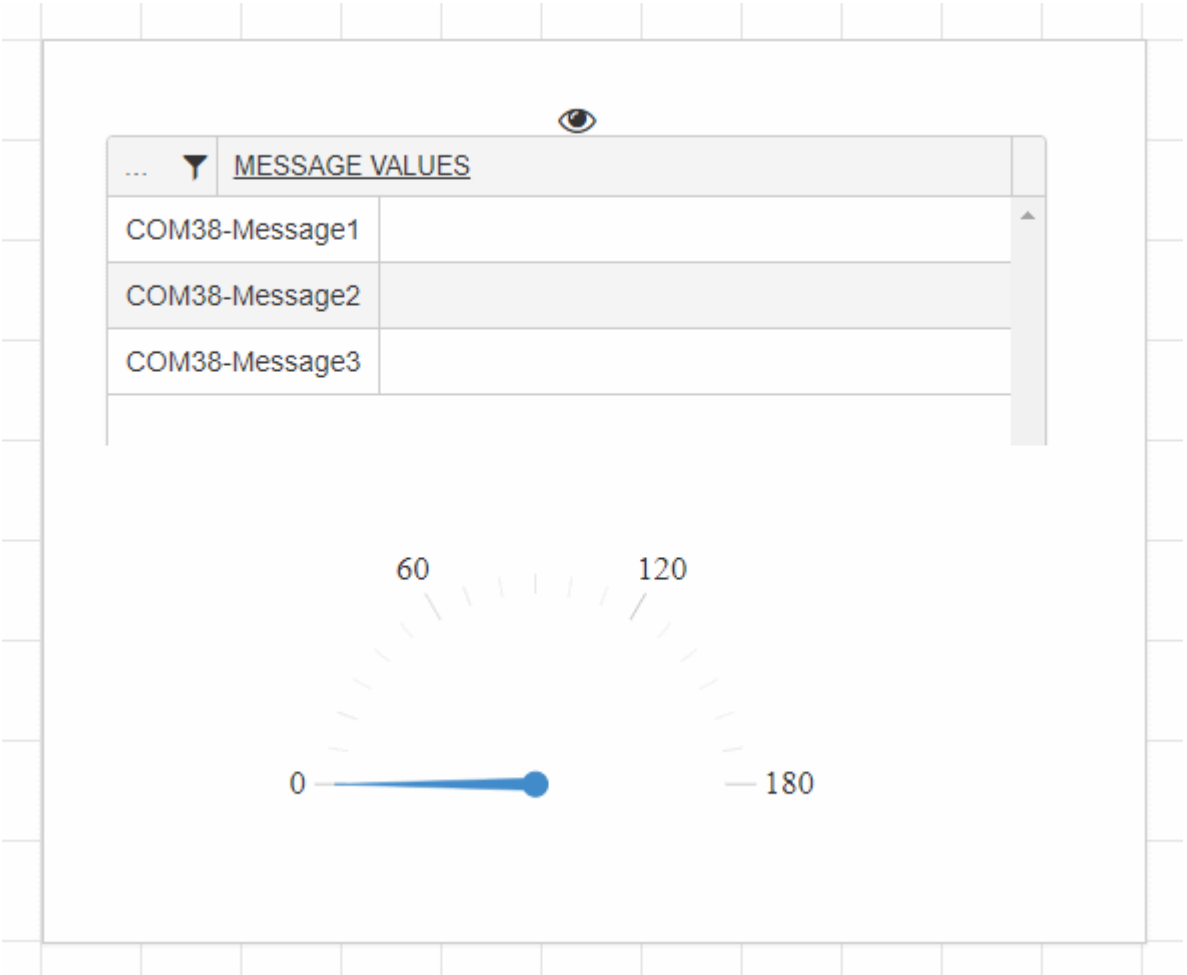
more Widgets together in a set position relative to each other. Not all widget types are eligible for being placed in a group however. Those that are are listed below:

- Gauges
  - Linear
  - Radial
  - Numeric
- Charts
- Real-Time Displays
- Manual Interfaces

There are a few other widget types which are managed on this page but are not eligible for inclusion in a Widget Group. They are listed below and covered in their own sections of the help documentation:

- Custom Messages
- Events
- Widget Groups (nested groups is not allowed)

Below is a visualization demonstrating these concepts



Here we see a Widget Group at run-time displayed on a Layout page. There is a single bounding box (the group) which contains two data visualization widgets - a Real Time Display and a Gauge

The group can be used to organize multiple, potentially related widgets (eg a Line Chart depicting Air Temperature along with a Numeric Gauge showing the current Air Temperature and an RTD showing a derived average over the last hour) together in a meaningful way.

Groups can also simply contain a single Widget if you choose, however as no widget can be directly added to a Layout they must be part of a group to be displayed at run time.

> Layouts are a wrapper around a set of Widget Groups and Widget Groups are a wrapper around a set of Widgets.

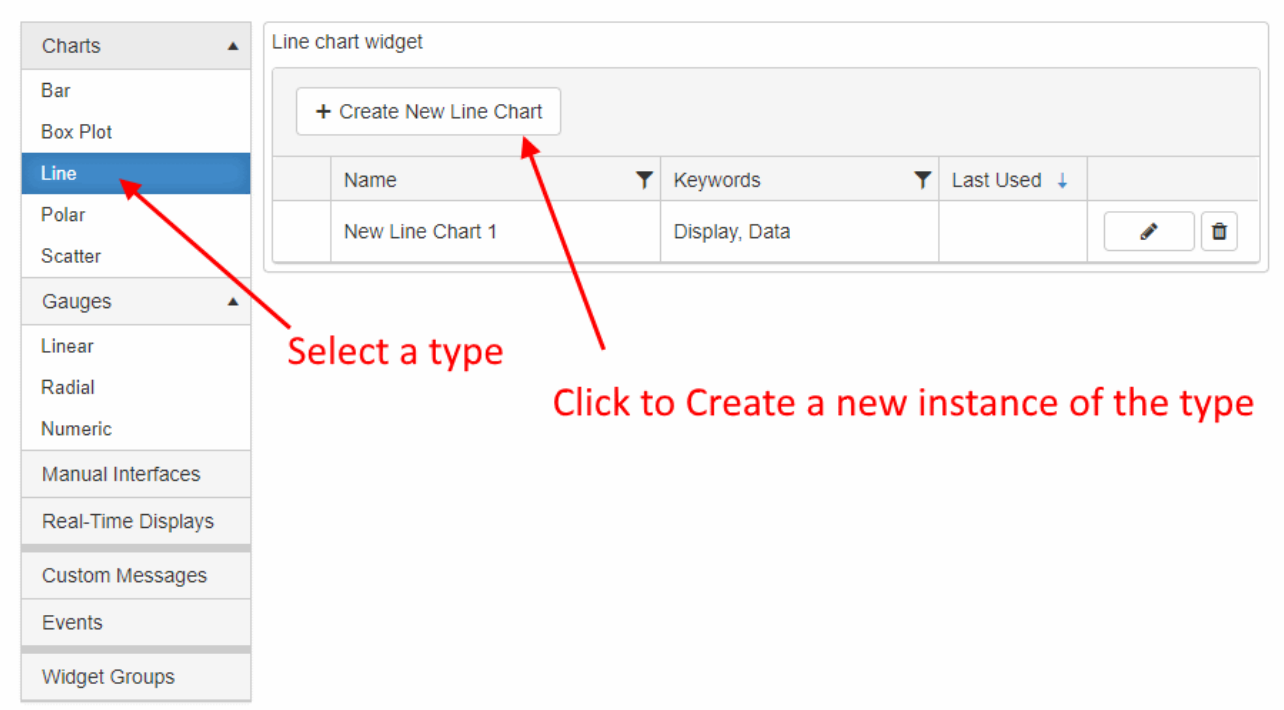
The same Widget Group shown above in design time makes this even clearer.

Creating Widgets

There are two ways to create a widget, you can do so via the *Layout Editor* or you can do so via the *Manage My Widgets* UI. Creation via the *Layout Editor* is covered in its portion of the help documentation and will not be covered here. Creation via the *Manage My Widgets* UI is very straightforward:

1. Browse to the UI via the *Template* link on the main menu or from the *Home Page* via the *Widgets* button (*Recently Used Layouts* block).

2. Select the type of widget you would like to create from the panel bar on the left.
3. Click the *+Create New button* at the top of the grid on the right.



4. Fill out the popup window with appropriate values and click *OK*
5. A blank template is created you'll see the the standard template layout with the editor targeting the type you had selected.

Editing Widgets

There are two ways to edit an existing widget, you can do so via the *Layout Editor* or you can do so via the *Manage My Widgets* UI. Editing via the *Layout Editor* is covered in its portion of the help documentation and will not be covered here. Editing via the *Manage My Widgets* UI is very straightforward:

1. Browse to the UI via the *Template* link on the main menu or from the *Home Page* via the *Widgets* button (*Recently Used Layouts* block).
2. Select the type of widget you would like to edit from the panel bar on the left.
3. Find the widget you want to change in the grid and click the *Edit* button (the one with the pencil).
4. You will see the standard template editor loaded with the widget you want to edit. Make your changes and don't forget to hit *Save*!

⚠ Changes made to widgets will potentially effect all running instances of it throughout the ship.

Deleting Widgets

You can only delete an existing widget via the *Manage My Widgets* UI.

1. Browse to the UI via the *Template* link on the main menu or from the *Home Page* via the *Widgets* button (*Recently Used Layouts* block).
2. Select the type of widget you would like to delete from the panel bar on the left.

3. Find the widget you want to delete in the grid and click the ***Delete*** button (the one with the trash can).

⚠ Be sure the widget is removed from any Widget Group referencing it or users will have gaps in their UI should they include those groups in a Layout

Management of Widget Groups via the *Manage My Widgets* UI is not as featured as doing so via the *Layout Editor*. If you want to create/edit Widget Groups it is best to do so via the *Layout Editor* where you have control over placement of individual widgets.

Charts

Charts are one of the more impactful data visualization widgets in terms of QA/QC and monitoring your sensor feeds. They allow you to display both live and historic data and also provide the ability to bundle up sequential values in boxes for quick review with minimal performance impact.

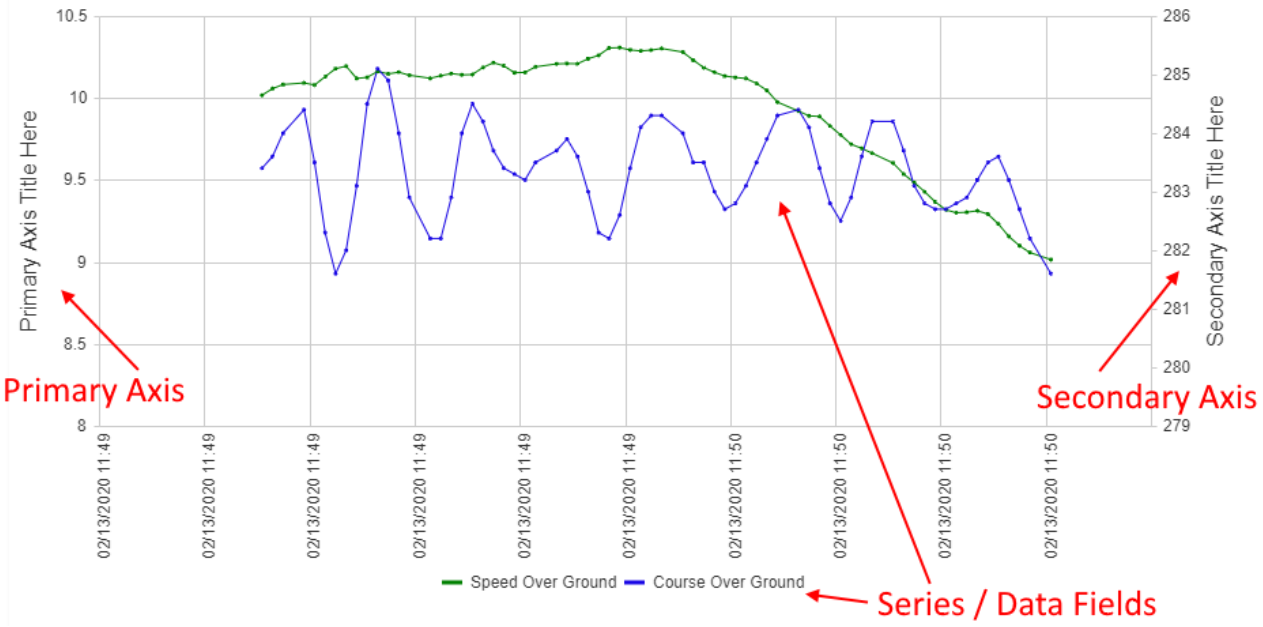
Designing the Widget

Click the ***Charts*** group in the *Manage My Widgets* panel then click the ***+Create New Chart*** button on top of the grid.

Enter the standard information for widget creation including a name, description, any keywords and permission level you wish this template to have and hit OK.

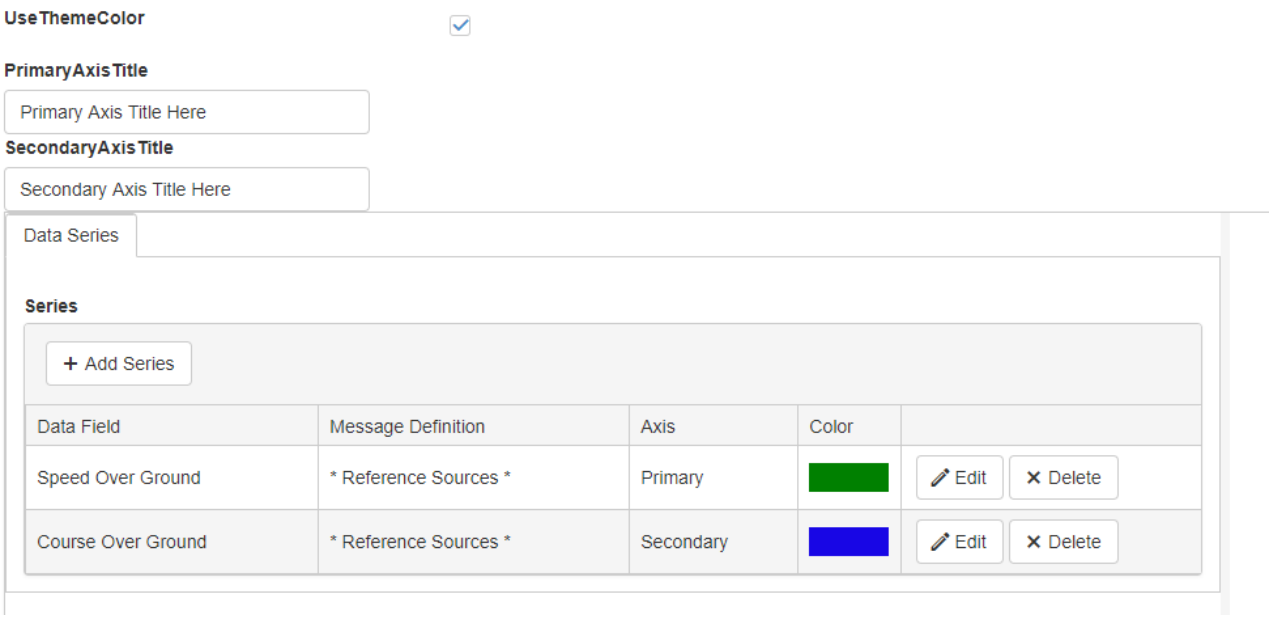
This will bring you to the editor for this widget.

Charts are so commonly used that not much explanation is required for you to understand the concept. Essentially each data feed you wish to display is setup as a Series. As values come in for a series they are put onto the screen with the vertical axis representing the value and the horizontal representing some other value (default is time). If the Chart is a Line chart then the sequential points are connected via a line segment. An optional second vertical axis can also be used if multiple series do not share a common range (eg SOG and COG, both are numeric but if displayed on the same axis then most likely the COG would flatten the SOG since it ranges all the way up to 360).



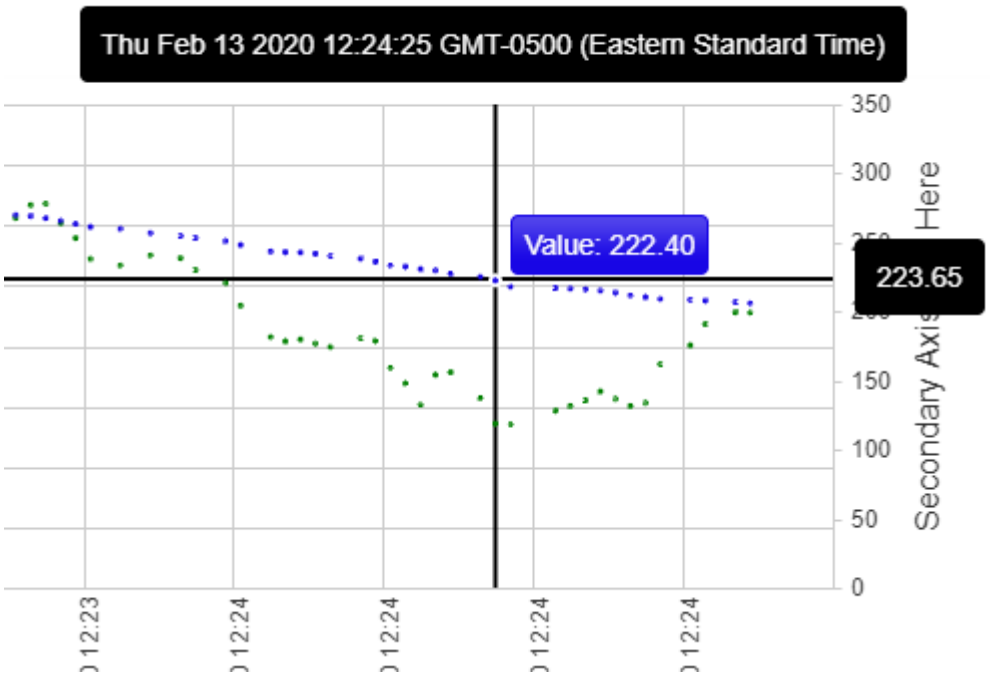
The Chart builder is very straightforward. Many of the options are available during run time instead (such as displaying as a box chart, a line chart or a scatter chart). You can choose to use the colors assigned via the selected SCS theme (this applies to the series as well), set the primary/secondary axis text and add 1 or more data feeds you wish to display.

> If one of your Series is on a secondary axis then that axis will be displayed at run time. If all of your Series are on the primary axis then the secondary axis will be hidden at run time.



Run Time

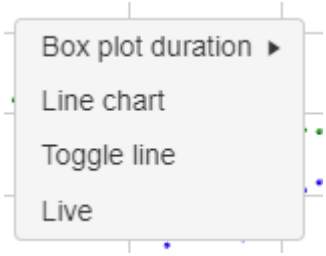
Once you have defined your Chart widget it can be added to a Widget Group and displayed on Layout.



As you move your cursor over the Chart you will notice cross hairs appear which tell you the value on each axis. Additionally, if you hover over a point, you can see the value for a given series under the mouse.

You can left click to "drag" your chart (when in run mode) left and right. This allows you to scroll back and forth in time to load historic data for your selected series.

You may use your mouse when to zoom in and out as well, this changes the total time frame being displayed. Be aware if you zoom out you may start to incur performance penalties as large quantities of data don't always play nice with web-based charting.



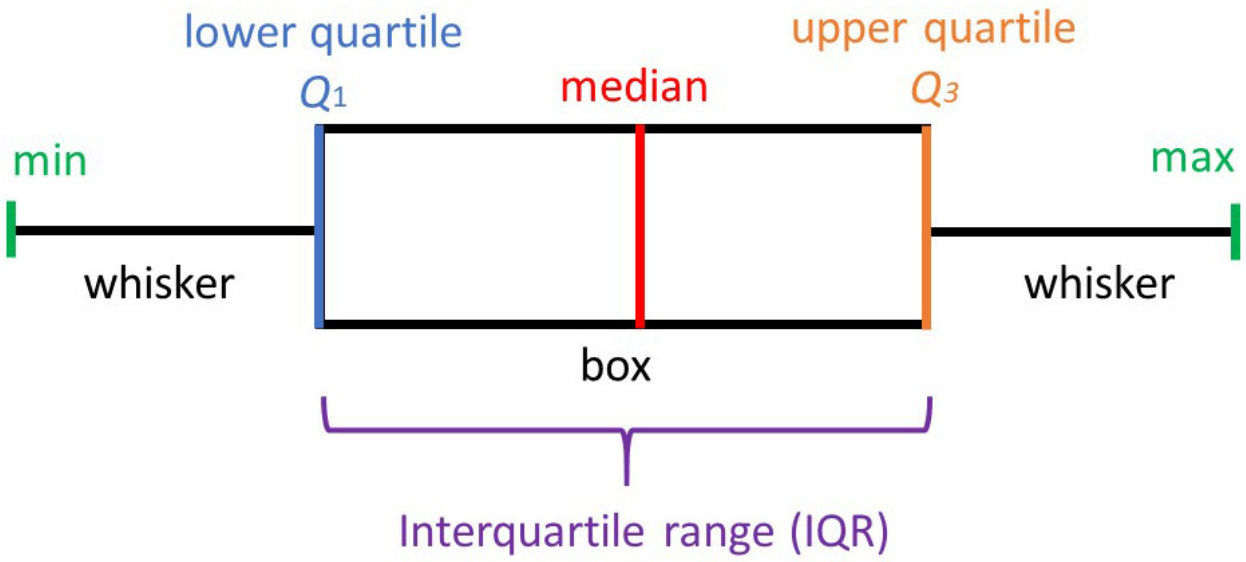
If you right click on the Chart you will be presented with a context menu which allows you to change some basic settings on your chart while it's running.

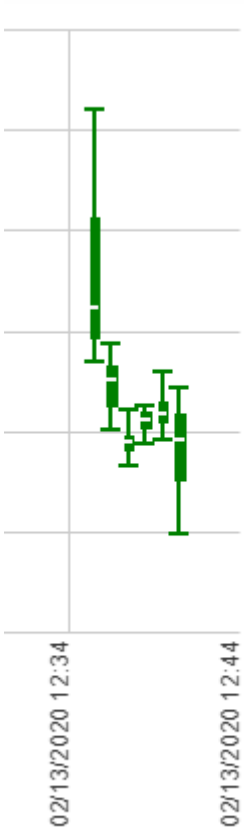
The first option lets you change your chart into a Box Plot

The second changes your Chart into a scatter/line chart. To see/hide the line be sure to click the *Toggle Line* option as well

To view live data (if you've scrolled into the past) click Live on the menu to go back to the current feed

Box Plots





The idea of Box Plots came about to resolve the long standing issue of a large number of data points bringing most charts to their knees in terms of performance. Instead, groups of datapoints (size of the group is determined by the Box pot duration value) are displayed as "Boxes". This method of presentation will allow you to see the general values of the series, including any outliers, without having to plot each individual point. More details (and the image above) can be found here - https://www.simplypsychology.org/boxplots.html

By plotting in boxes, you can quickly see the general trend of your series over time and should any min/max values appear to be erroneous you can zoom into them to view the scatter/line chart instead for finer details.

If you hover your mouse over a box you will get a tooltip showing you the data making up the box (eg your min/max, etc)

Linear Gauges

Gauges provide the ability to display real time data feeds coming from ACQ on your user interface. You can only display distinct data items (data fields) as full RAW messages are usually not a single numeric value. Gauges come in 3 flavors: Linear, Radial and Numeric. They are very useful for illustrating a progress towards a goal, the current status of some value within a range of upper and lower bounds, or a summary of some fluctuating metric.

Linear Gauges represent data on a linear scale.

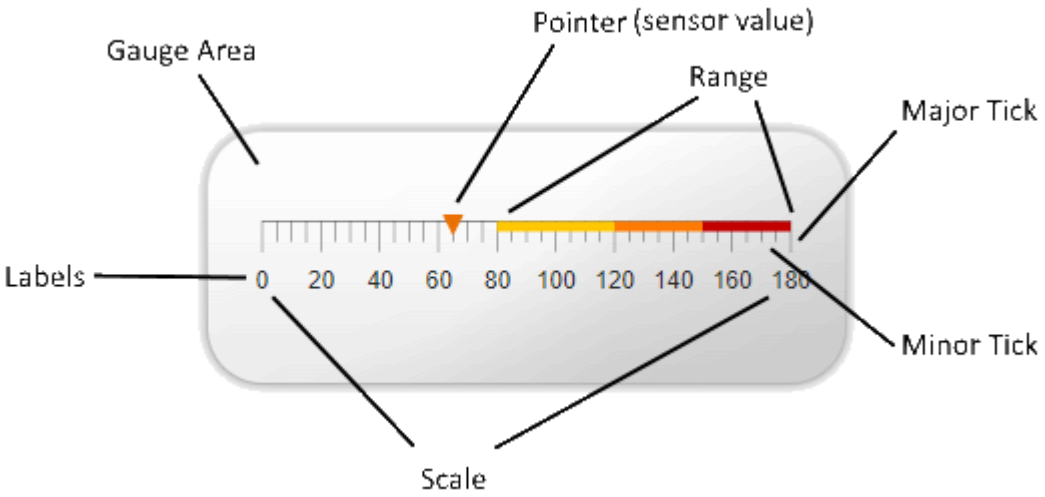
Designing the Widget

Click the ***Linear*** link under the *Gauges* group in the *Manage My Widgets* panel then click the *+Create New* ***Linear Gauge*** button on top of the grid.

Enter the standard information for widget creation including a name, description, any keywords and permission level you wish this template to have and hit OK.

This will bring you to the editor for this widget.

This image indicated the major components you can edit, details regarding each can be found in the following sections.

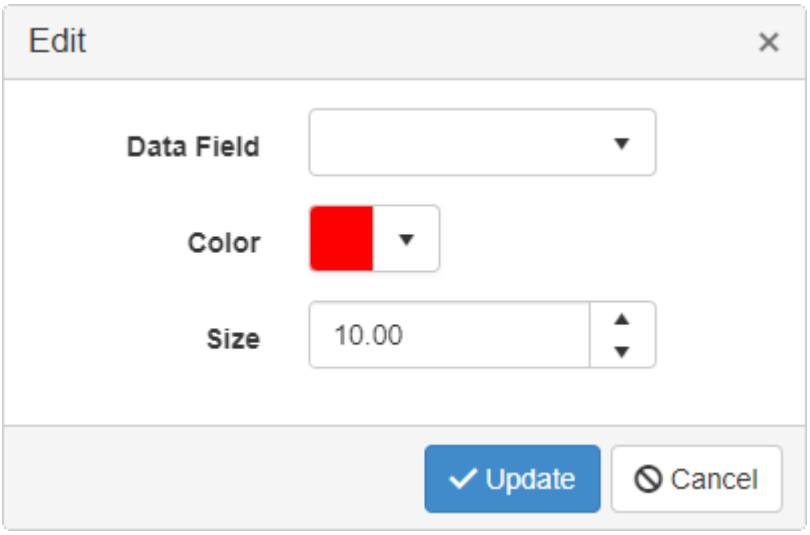


The editor for linear gauges is mainly comprised of a tab strip with 5 tabs:

Pointers

The *Pointers* tab is where you define the actual indicator which moves on the gauge to display the current value of the underlying sensor item. You must have at least one pointer on your gauge for it to make any sense, multiple pointers are supported if you have many datafields you would like displayed at the same time on the same gauge.

Each pointer represents a single real time data item. To add a pointer to your gauge click the *+Add Pointer* button located at the top of the pointer grid.



You will be presented with the editor for a pointer.

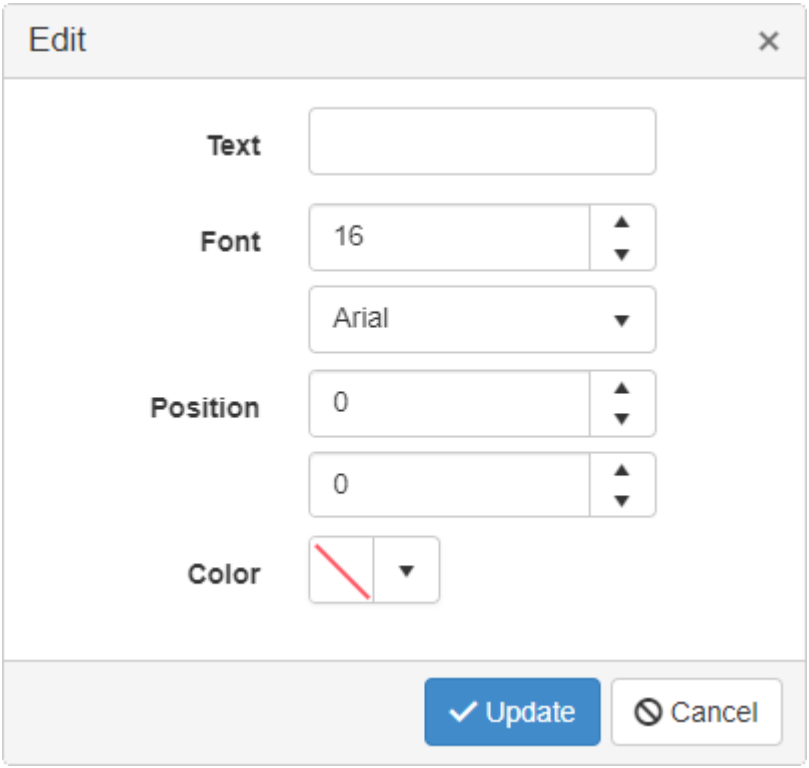
At the top you will see the option to select the source of the pointer data. Whatever datafield you select here will be what the pointer shows when the gauge is running.

If you have chosen to not use a theme (it is recommended you always use a theme) you may specify an explicit color to use for your pointer.

Use the *Size* setting to increase or decrease the displayed size of the pointer.

Custom Labels

The gauge allows you to integrate zero or more custom text elements. While some gauges may make sense on their own or inside a larger custom widget, most gauges would need a label of sorts so the user knows what exactly they're looking at. For instance a circular gauge could be representing a COG, Wind Direction or any other number of data sources. If a user just casually looks at the gauge they will know what it's current value is, but will have no idea what "it" is. Adding a label will help clarify what your gauge represents. It's recommended to add at least one label with the name of the pointer source or a higher level descriptor (eg "True Wind"). To add a label to your gauge click the ***+Add Custom Label*** button located at the top of the custom labels grid.



You will be presented with the editor for a custom label.

The First input is where you specify the actual content / text of the label

The Font will allow you to change the size and selected font of the label

The position allows you to specify the offset (percentage) of the label. The first input is horizontal offset (eg a value of 50 would mean the label starts min the middle of the screen)

The second input is the vertical offset (eg a value of 50 would mean the label starts halfway down the screen)

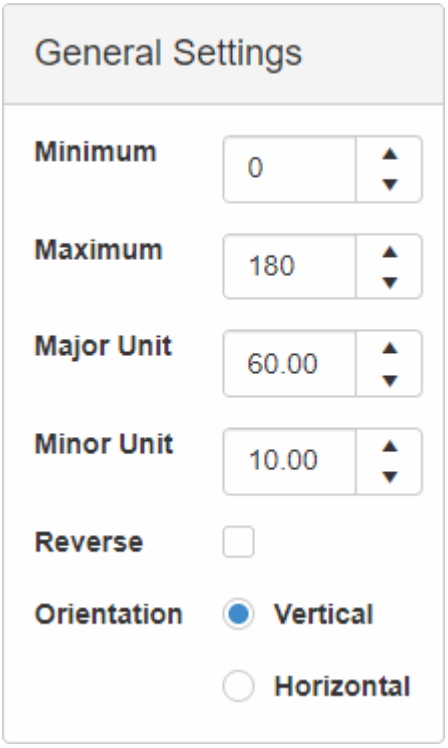
If you have chosen to not use a theme (it is recommended you always use a theme) you may specify an explicit color to use for your pointer.

Gauge Area

The *Gauge Area* tab is where you define settings related to the frame and background of the overall gauge. The border is used to define a wrapper which goes around the gauge. The margin is used to specify spacing around the gauge, the larger the number to more 'buffer space' you will see in that area of the gauge.

Scale

The *Scale* tab is where you define settings related to the scale associated with the pointer. It has a few subsections which can be used to refine details of your display.



The *General Settings* section allows you to specify a most of the primary details relevant to making your display relevant and useful. Namely:

Minimum: The lowest numeric value the pointer can go to.

Maximum: The highest numeric value the pointer can go to.

The gauge has both major and minor tick marks which help the user know the numeric value the pointer is current set to.

Major Unit: A 'major' indicator will appear every 'x' values of the data fields unit.

Minor Unit: A 'minor' indicator will further divide up the 'major' units by placing a smaller indicator every 'x' values.

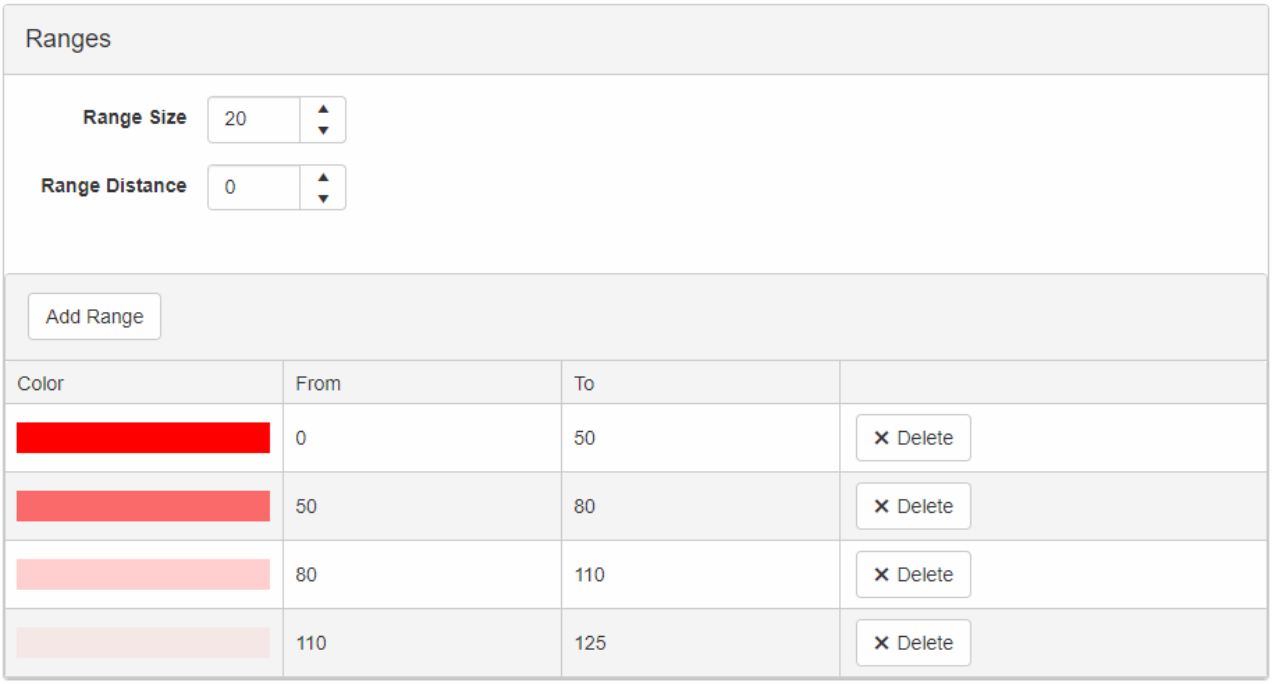
Reverse: Flip the scale 180 degrees (eg instead of 0 to 180 it would be 180 to 0)

Orientation: Determines if the gauge should be displayed on a horizontal (long) axis or a vertical (high) axis.

Major and *Minor* tick marks are pretty self explanatory. They help the user take the visual position of the pointer and correlate that to a numeric value. It is advisable to have your major and minor units be factors of the difference in the scale for symmetry purposes. You can change the tick mark sizes, color or even hide them altogether.

Labels are the numeric values being displayed on the scale which the pointers point to. As with the other components you can change their position, size, color, spacing and various other aesthetic properties.

Range

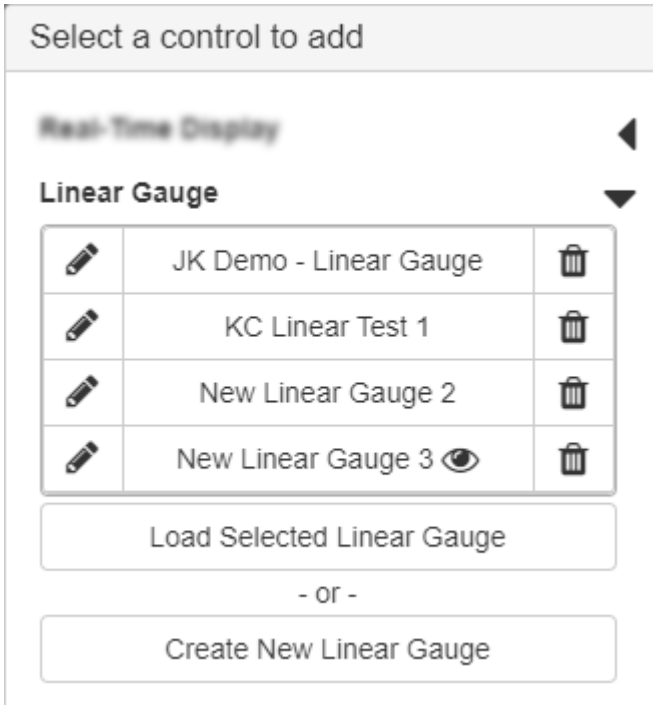


Ranges are colorful indicators which provide feedback on the current position and value of the pointer/ sensor. In most cases this is used to indicate either an 'ideal' range (eg pointer should strive to be inside the min/max of the range) or a progressive state towards good/bad (such as a green to red gradient).

To build a range you must specify one or more colors which should be displayed with a start/end value. If you wish to apply a gradient you must do so gradually as seen in the image on the left.

Run Time

Linear Gauges are, like other widgets, added to a layout via a Widget Group.



Simply find the Linear Gauge node and expand it to see a list of all Linear Gauge templates available to you. Create a new one if needed or load a previously defined one into your group.

The Gauge you select should start and the pointer(s) will reflect the current value(s) of the underlying sensor data each is associated with.

Numeric Gauges

Gauges provide the ability to display real time data feeds coming from ACQ on your user interface. You can only display distinct data items (data fields) as full RAW messages are usually not a single numeric value. Gauges come in 3 flavors: Linear, Radial and Numeric. They are very useful for illustrating a progress towards a goal, the current status of some value within a range of upper and lower bounds, or a summary of some fluctuating metric.

Numeric Gauges represent data as a changing numeric value, similar to an old LCD clock.

Designing the Widget

Click the ***Numeric*** link under the *Gauges* group in the *Manage My Widgets* panel then click the *+Create* ***New Numeric Gauge*** button on top of the grid.

Enter the standard information for widget creation including a name, description, any keywords and permission level you wish this template to have and hit OK.

This will bring you to the editor for this widget.

The editor for numeric gauges is mainly comprised of a tab strip with 2 tabs:

Data Field

The *Data Field* tab is where you define the source of the numeric data you wish to display with this gauge. Unlike the other gauge types, numeric gauges only support a single source.

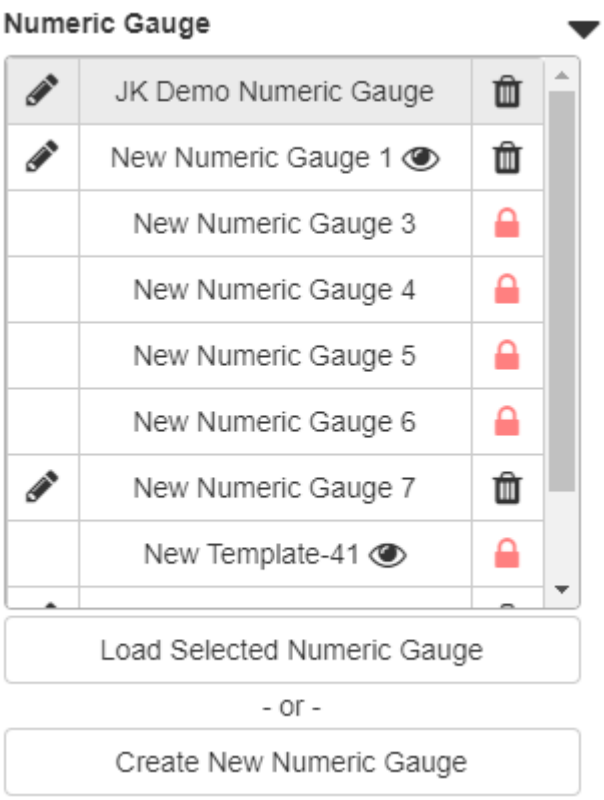
Gauge Display

The *Gauge Display* tab is where you define settings related to the display of the numeric value. You can specify a minimum and/or maximum value, set the color, size, etc of the font and set the method of numeric display. This method must be one of the following:

Fixed Decimal	Locks in the number of digits after the decimal point
Precision	Locks in the total number of digits displayed
Exponetial	Uses the scientific / exponential notation to display the value and sets the number of digits after the decimal point (useful for LARGE numbers).

Run Time

Numeric Gauges are, like other widgets, added to a layout via a Widget Group.



Simply find the Numeric Gauge node and expand it to see a list of all Radial Gauge templates available to you. Create a new one if needed or load a previously defined one into your group.

The Gauge you select should start and a numeric readout will reflect the current value(s) of the underlying sensor data the Gauge is associated with.

Radial Gauges

Gauges provide the ability to display real time data feeds coming from ACQ on your user interface. You can only display distinct data items (data fields) as full RAW messages are usually not a single numeric value. Gauges come in 3 flavors: Linear, Radial and Numeric. They are very useful for illustrating a progress towards a goal, the current status of some value within a range of upper and lower bounds, or a summary of some fluctuating metric.

Radial Gauges represent data on a linear scale.

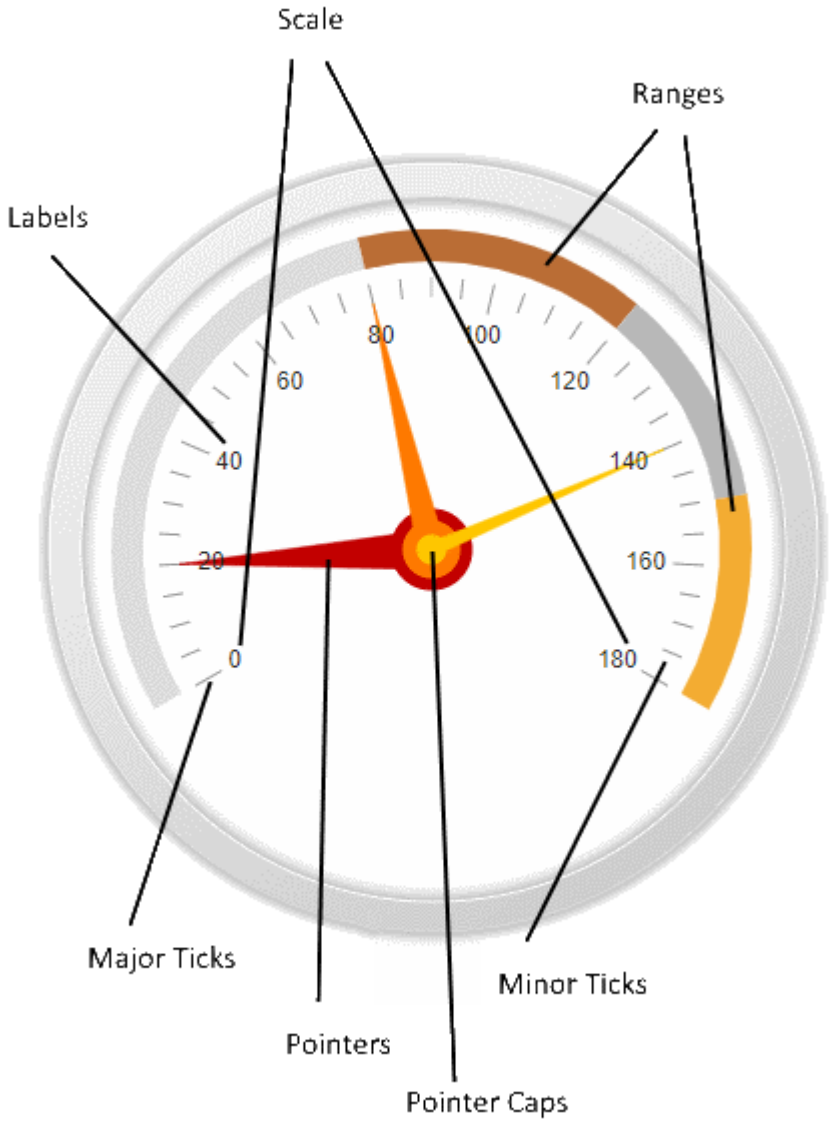
Designing the Widget

Click the ***Radial*** link under the *Gauges* group in the *Manage My Widgets* panel then click the *+Create New* ***Radial Gauge*** button on top of the grid.

Enter the standard information for widget creation including a name, description, any keywords and permission level you wish this template to have and hit OK.

This will bring you to the editor for this widget.

This image indicated the major components you can edit, details regarding each can be found in the following sections.

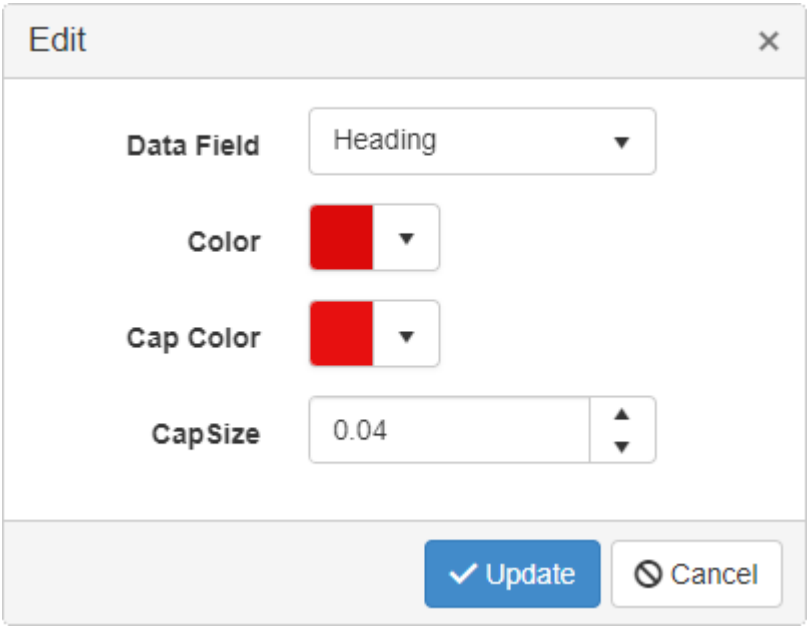


The editor for radial gauges is mainly comprised of a tab strip with 5 tabs:

Pointers

The *Pointers* tab is where you define the actual indicator which moves on the gauge to display the current value of the underlying sensor item. You must have at least one pointer on your gauge for it to make any sense, multiple pointers are supported if you have many datafields you would like displayed at the same time on the same gauge.

Each pointer represents a single real time data item. To add a pointer to your gauge click the *+Add Pointer* button located at the top of the pointer grid.



You will be presented with the editor for a pointer.

At the top you will see the option to select the source of the pointer data. Whatever datafield you select here will be what the pointer shows when the gauge is running.

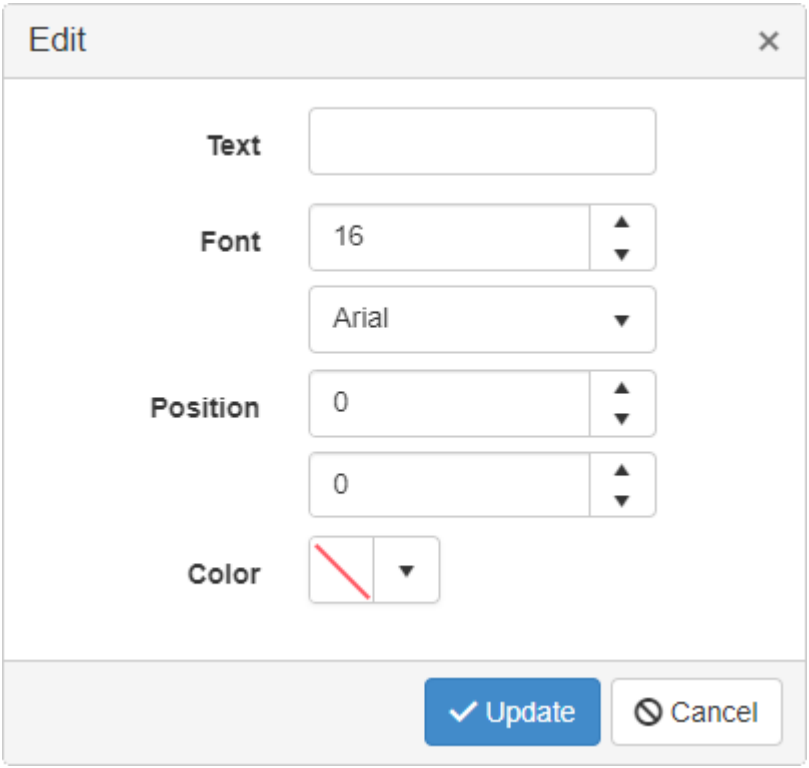
If you have chosen to not use a theme (it is recommended you always use a theme) you may specify an explicit color to use for your pointer.

Use the *Cap Color* setting to change the color of the Cap for the given pointer

Use the *Cap Size* setting to increase or decrease the displayed size of the Cap for the pointer.

Custom Labels

The gauge allows you to integrate zero or more custom text elements. While some gauges may make sense on their own or inside a larger custom widget, most gauges would need a label of sorts so the user knows what exactly they're looking at. For instance a circular gauge could be representing a COG, Wind Direction or any other number of data sources. If a user just casually looks at the gauge they will know what it's current value is, but will have no idea what "it" is. Adding a label will help clarify what your gauge represents. It's recommended to add at least one label with the name of the pointer source or a higher level descriptor (eg "True Wind"). To add a label to your gauge click the ***+Add Custom Label*** button located at the top of the custom labels grid.



You will be presented with the editor for a custom label.

The First input is where you specify the actual content / text of the label

The Font will allow you to change the size and selected font of the label

The position allows you to specify the offset (percentage) of the label. The first input is horizontal offset (eg a value of 50 would mean the label starts min the middle of the screen)

The second input is the vertical offset (eg a value of 50 would mean the label starts halfway down the screen)

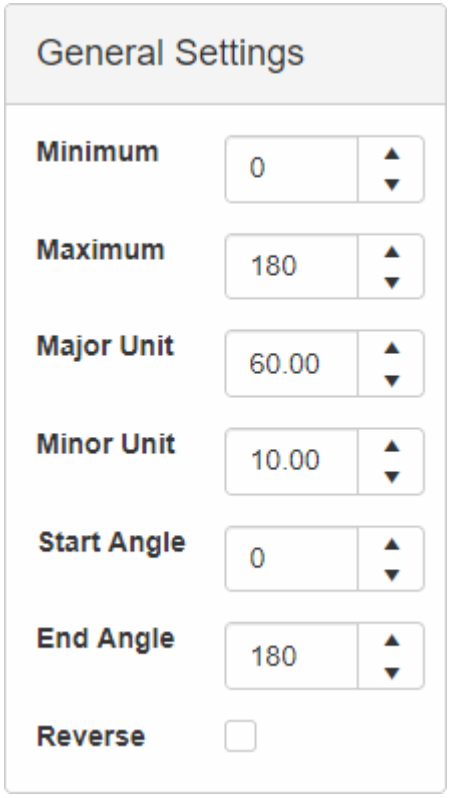
If you have chosen to not use a theme (it is recommended you always use a theme) you may specify an explicit color to use for your pointer.

Gauge Area

The *Gauge Area* tab is where you define settings related to the frame and background of the overall gauge. The border is used to define a wrapper which goes around the gauge. The margin is used to specify spacing around the gauge, the larger the number to more 'buffer space' you will see in that area of the gauge.

Scale

The *Scale* tab is where you define settings related to the scale associated with the pointer. It has a few subsections which can be used to refine details of your display.



The *General Settings* section allows you to specify a most of the primary details relevant to making your display relevant and useful. Namely:

Minimum: The lowest numeric value the pointer can go to.

Maximum: The highest numeric value the pointer can go to.

The gauge has both major and minor tick marks which help the user know the numeric value the pointer is current set to.

Major Unit: A 'major' indicator will appear every 'x' values of the data fields unit.

Minor Unit: A 'minor' indicator will further divide up the 'major' units by placing a smaller indicator every 'x' values.

Start Angle: The angle the start of the arc begins at.

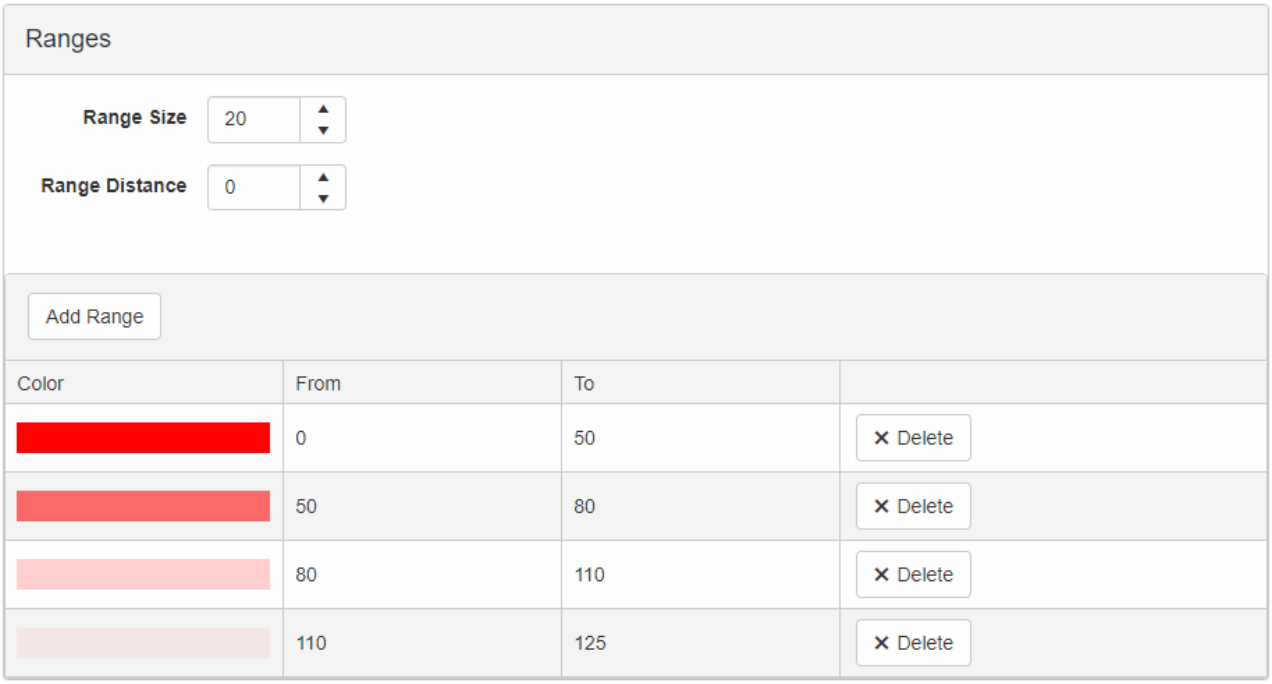
End Angle: The angle the end of the arc terminates at.

Reverse: Flip the scale 180 degrees (eg instead of 0 to 180 it would be 180 to 0)

Major and *Minor* tick marks are pretty self explanatory. They help the user take the visual position of the pointer and correlate that to a numeric value. It is advisable to have your major and minor units be factors of the difference in the scale for symmetry purposes. You can change the tick mark sizes, color or even hide them altogether.

Labels are the numeric values being displayed on the scale which the pointers point to. As with the other components you can change their position, size, color, spacing and various other aesthetic properties.

Range

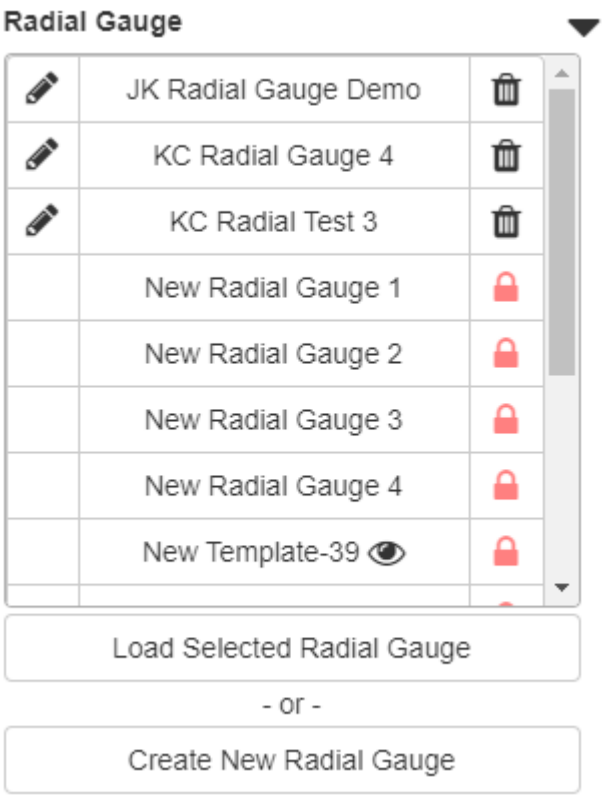


Ranges are colorful indicators which provide feedback on the current position and value of the pointer/ sensor. In most cases this is used to indicate either an 'ideal' range (eg pointer should strive to be inside the min/max of the range) or a progressive state towards good/bad (such as a green to red gradient).

To build a range you must specify one or more colors which should be displayed with a start/end value. If you wish to apply a gradient you must do so gradually as seen in the image on the left.

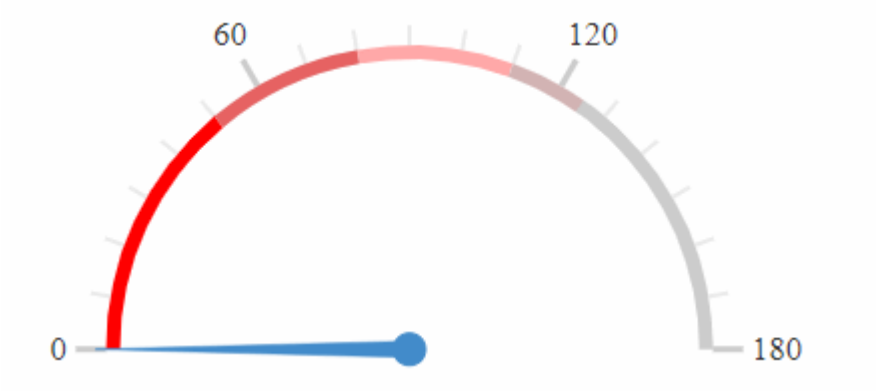
Run Time

Radial Gauges are, like other widgets, added to a layout via a Widget Group.



Simply find the Radial Gauge node and expand it to see a list of all Radial Gauge templates available to you. Create a new one if needed or load a previously defined one into your group.

The Gauge you select should start and the pointer(s) will reflect the current value(s) of the underlying sensor data each is associated with.



Real Time Displays

Real Time Displays provide the ability to display real time data feeds coming from ACQ on your user interface. You can display RAW data exactly as it comes out of the sensor and/or parsed data if you are

interested in the data fields themselves. Real Time Displays are extremely simple to setup and are the primary widgets to use when you wish to to view data streams as text.

Designing the Widget

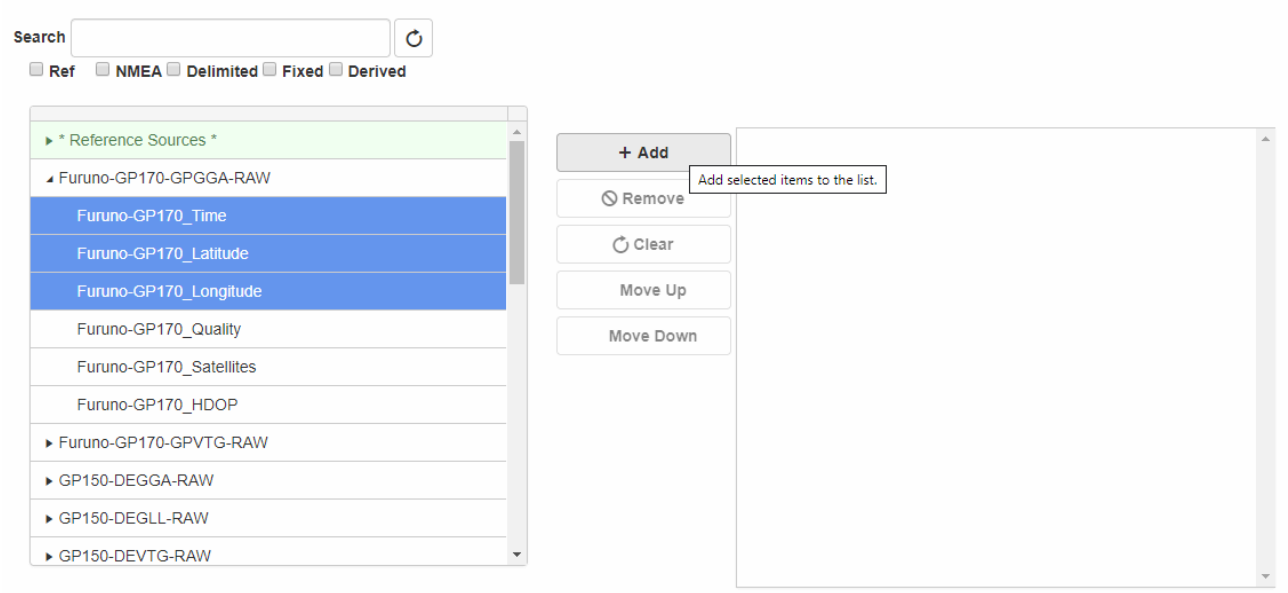
Click the ***Real-Time Displays*** link on the *Manage My Widgets* panel then click the *+Create New Real-Time* ***Display*** button on top of the grid.

Enter the standard information for widget creation including a name, description, any keywords and permission level you wish this template to have and hit OK.

This will bring you to the editor for this widget.

The editor is broken down into 3 columns. On the left you will find the standard sensor selection screen, in the middle you will see the command buttons and on the right you will see all the sensors you wish to display on your UI in this widget.

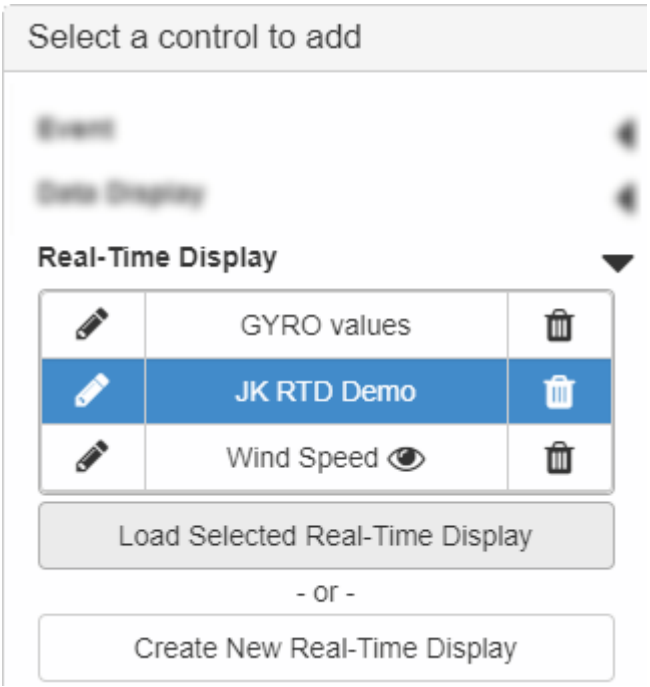
Simply pick one or more messages/data-fields on the left, click the *+Add* button to add them to your display and change their order on the right as you see fit.



The current data for each of the items in the right hand list will be displayed, in the order as you see it, on the widget when it is run. You can build logical groupings (eg all GYROs in one display, or the relative winds and true winds in one display, or all data needed by the bridge, etc) and create as many real time display widgets as you need to accommodate your various QA/QC and data visualization goals.

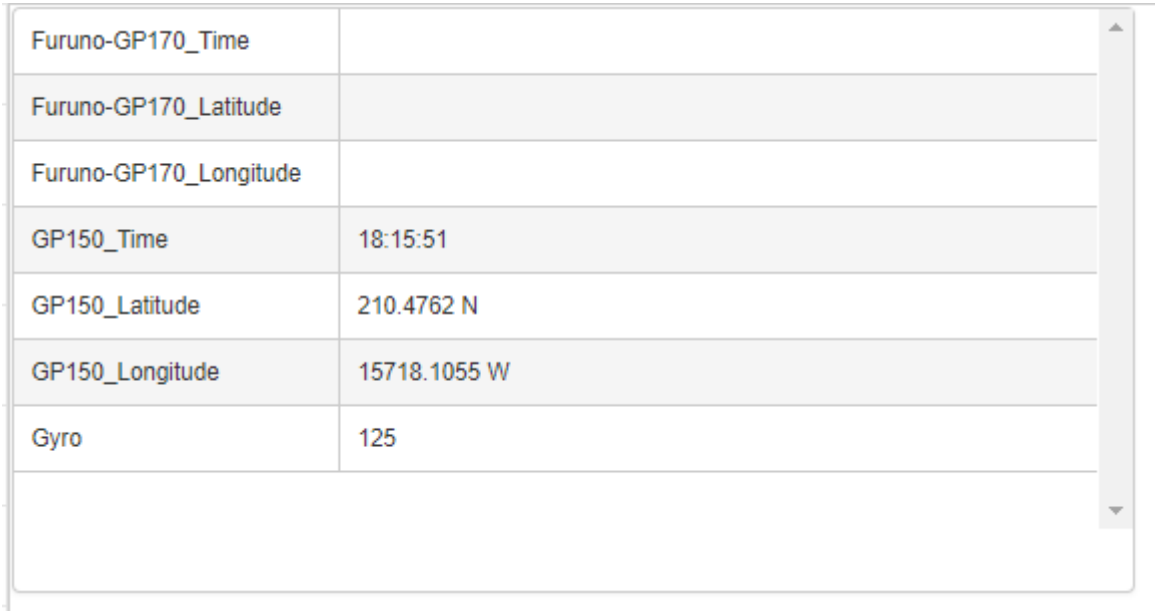
Run Time

Real Time Displays are, like other widgets, added to a layout via a Widget Group.



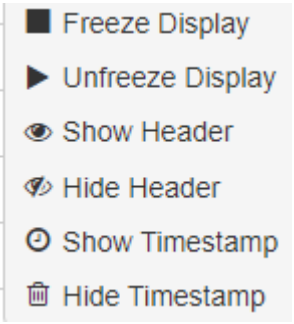
Simply find the Real-Time Display node and expand it to see a list of all RTD templates available to you. Create a new one if needed or load a previously defined one into your group.

Each of the items you added to your RTD template will start to update itself with it's current value as supplied by ACQ.



Furuno-GP170_Time	
Furuno-GP170_Latitude	
Furuno-GP170_Longitude	
GP150_Time	18:15:51
GP150_Latitude	210.4762 N
GP150_Longitude	15718.1055 W
Gyro	125

Right clicking on the RTD provides a context menu allowing for different actions to be performed.



Manual Interfaces

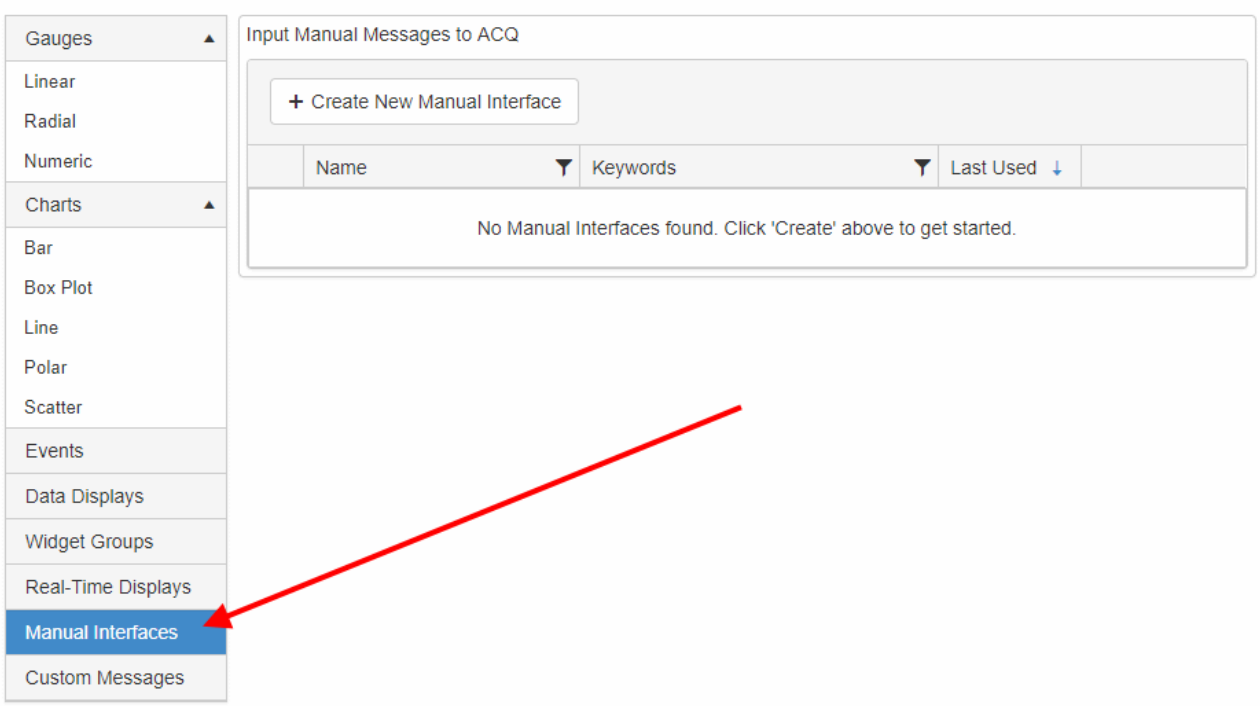
Manual Interfaces are a type of Sensor Interface built inside *CFE* which allow a human to provide manual data to the data acquisition service as if the human themselves were a sensor. Times this may be useful are when normal sensors aren't available to record the data, such as cloud coverage, mammal sightings, etc.

While Manual Interfaces are built via *CFE* by an SCS Administrator, they can be utilized on a Layout the same as most other templated widgets. Putting a Manual Interface widget on a Layout allows users to act as the manual data feed supplying the device. In other words, this widget is the way you provide manual data and send it to ACQ.

See also - Manual Interfaces

Designing the Widget

Creation of this widget is extremely simple, however before you can do so you must have at least 1 Manual Interface defined in *CFE*.
Click the *Manual Interfaces* link on the *Manage My Widgets* panel then click the *+Create New Manual Interface* button on top of the grid.



Enter the standard information for widget creation including a name, description, any keywords and permission level you wish this template to have and hit OK.

This will bring you to the editor for this widget.

> ⚠ Anyone who can access this widget, even if set to read-only, can send data to ACQ on behalf of the Manual Interface this widget represents!

The editor only has 1 real option as most the details are defined in CFE.

There is a single checkbox which, if checked, automatically clears the input areas for your next entry when you click 'send' to submit your manual data the system. If left unchecked the your data remains exactly as you type it and you can submit the same multiple times.

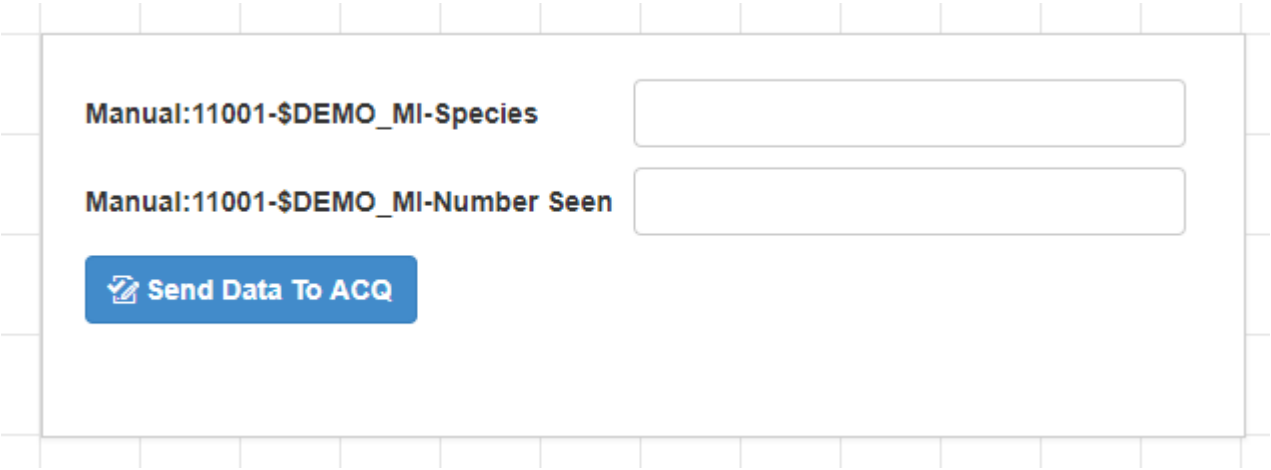
> If your data is somewhat unique each time or you want to ensure you manually retype/refresh your data prior to sending then leave this checked. If your data is more complicated (many datafields) and not much changes you might want to uncheck this so you don't have to retype everything each time.

The only other option is to select which Manual Interface you want this widget to represent.

Run Time

When running this widget the user will see a line for each data field defined by the Manual Interface Message Definition inside CFE.

The user may enter a value for each data field and when happy with their submission click the *Send Data to ACQ* button located at the bottom of the widget.



This effectively sends the data to ACQ and it will be treated the same as other sensor data (timestamped and pushed to any referencing events, logged to the database and files, send over UDP, etc). If the Clear inputs after sending checkbox was checked, all user entered values will be reset so the user can enter fresh values upon their next submission.

Custom Messages

Custom Messages provide the ability to transmit collected sensor data in user defined formats over various outputs ranging from serial/network communication ports to files or emails. Custom Message widgets are similar to Events in that they are not utilized inside ***Layouts*** and instead have their own dedicated management page. The user defines the formatted message and data to be sent through a Custom Message widget and then runs them via the *Custom Messaging* page under the *Data Management* menu item.

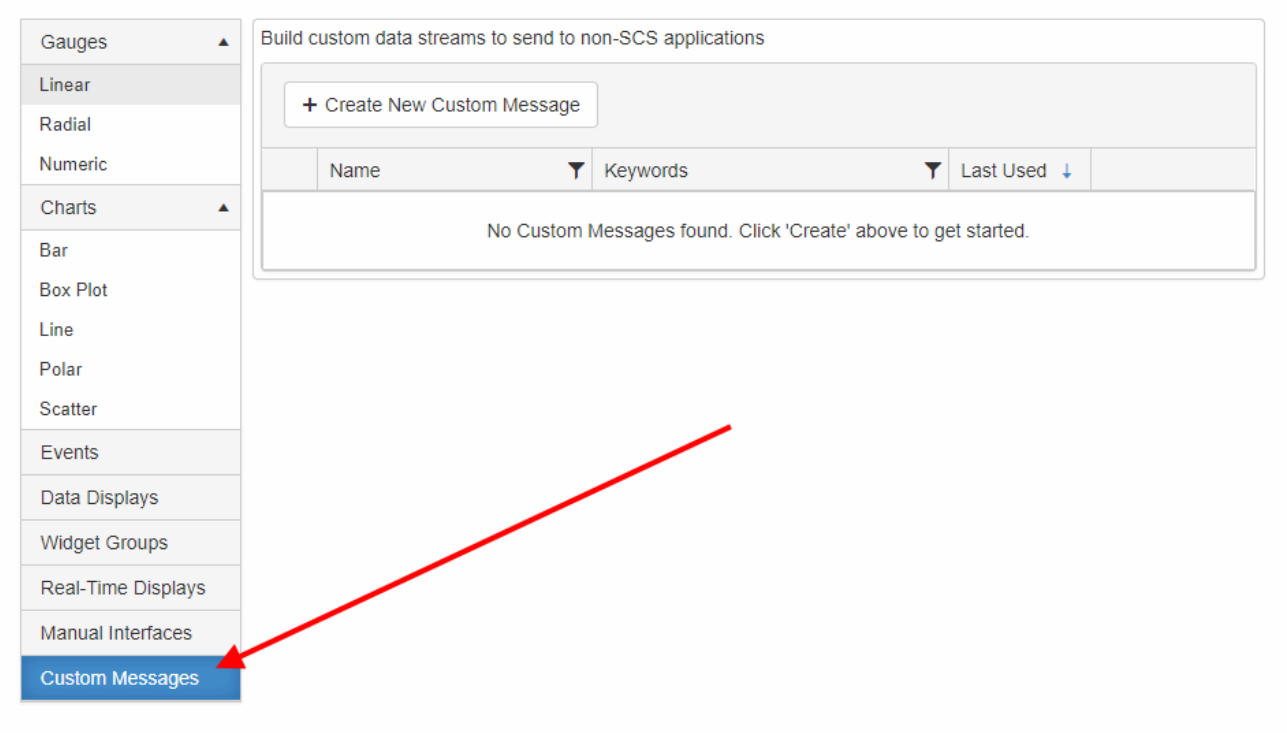
Custom Messages are very useful to combine various sensor data items into a single data string which you can then send to other programs at a periodic interval. For instance, if a scientist has a tool which requires GPS, both Gyros and the current depth, you can create a Custom Message with all these components and send them to the tool every second over the network. In the past this feature of SCS has been critical for 3rd party software running on the ship to get data streams quickly and efficiently.

This feature is managed by the Data Dissemination Service running in the background.

> ⚠ As with all of SCS - no critical (Life or Limb) systems should rely on this feature. These type systems should be implemented per vendor guidance.

Designing the Widget

Click the ***Custom Messages*** link on the *Manage My Widgets* panel then click the *+Create New Custom* ***Message*** button on top of the grid.



Enter the standard information for widget creation including a name, description, any keywords and permission level you wish this template to have and hit OK.

This will bring you to the editor for this widget.

The editor is broken down into three main areas of concern,

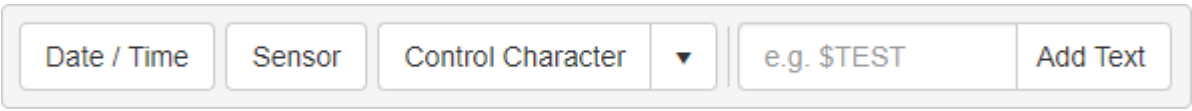
1. Building out the contents of the data you want to send
2. How you want to package those contents
3. How often and where you want to send the custom package

Message Definition

The *Message Definition* section is where you build out the contents of the data you want to send. It consists of a toolbar where you pick what you want to add to your message and then a grid showing you each component you have added and how it relates to the others (e.g. display order). You can change the order of items in the grid by drag-and-drop.

> As you add components or change your packaging format a sample of your message will be displayed for your review at the bottom of the editor.

The toolbar consists of 4 main types of components you can add to your message.



Date / Time

The first is a *Date / Time*, if you want to add a timestamp to your outgoing message this is the option you should choose. SCS provides a large number of variations for formatting a date and or time, choose the one that is right for the receiving application.

> As an alternative, you could use a timestamp data field from a sensor instead, such as a ZDA message, though you cannot customize their format.

> SCS Timestamps are taken from the SCS server's local system clock (should be sync'd with the ship's time/NTP/PTP/AD) and by default is in UTC.

> Though in no way required, it is recommended that the first value in your message be some sort of ID which uniquely identifies your package (use the *Add Text* component to do this). That can be immediately followed by the timestamp of the message. Take a look at the $GPGGA message as an example.

Sensor

The next component option is *Sensor*, this is where the majority of your message definition will come from. Clicking this allows you to add sensor data to your message, this can be a full RAW message definition or an individual data field. You will most likely click this button repeatedly to add multiple sensor data components to your message.

Clicking this will bring up the standard sensor selection UI, click the item you want to include in your message and hit OK.

> If the sensor item you are adding is a data field and is in the Latitude or Longitude category (CFE) then you can also reformat it into various Lat/Lon formats (such as Decimal Degrees) on the fly prior to sending it out via this Custom Message.

Control Character

The *Control Character* component adds a control character (a hidden character which indicates something to the computer reading the message) to your definition. The two most common control characters (a line feed and a carriage return) are quickly available via the dropdown. A full list is available if you click the button itself.

In order for the receiving tool to know that it has received a full message it is advisable to have a control character at the end of your definition. Normally this would be a carriage return followed by a line feed. Work with anyone receiving (the person who requested this message) the data to determine what, if any, control characters they need and where they need them.

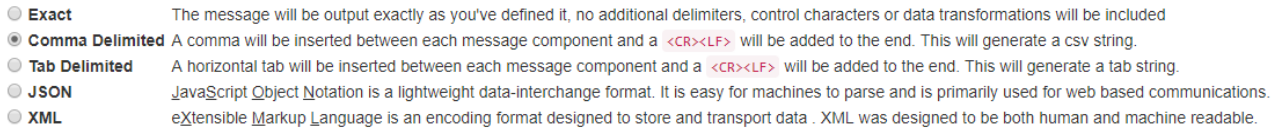
> ⚠ Most likely the only time you need to add control characters is if you choose *Exact* for your packing format. If you choose any format other than *Exact* chances are you can ignore control characters completely and not add them to your message at all as it is automatically taken care of for you.

Custom Text

The Add Text component allows you to enter hard coded custom strings into your message, such as a unique ID or custom sentence label at the beginning. If you were to choose the Exact packaging format then you might use this to add commas between each component. Essentially whatever you type here will show up in your message exactly as you typed it.

Packaging Format

The ***Packaging Format*** section is where you decide how you want SCS to combine all your message components prior to sending it out. There are a few options, talk with the owners of the receiving applications to figure out which works best for them.



Each option has a description as to how it would package your message. The most likely option is *Comma Delimited*, which injects a comma between each component in your message and automatically appends a carriage return and a line feed to the end for you.

For example, for the given component list below:

Component Type	Value	
Custom Text	$DEMO_CM	Edit Delete
Date / Time	SCS Date & Time - Short	Edit Delete
Sensor	Furuno-GP170_Latitude	Edit Delete
Sensor	Furuno-GP170_Longitude	Edit Delete
Sensor	Pressure	Edit Delete

A *Comma Delimted* format results in a clean string

$DEMO_CM,2019-08-23 18:33:40Z,-Unknown-,-Unknown-,-Unknown- <cr><lf>

> ⚠ ***-Unknown-*** is injected when sensor data for the given component is not available (ACQ is off, sensor is off, etc)

But if changed to the *Exact* format then the commas are lost and no control characters are added for you

$DEMO_CM2019-08-23 18:39:06Z-Unknown--Unknown--Unknown-

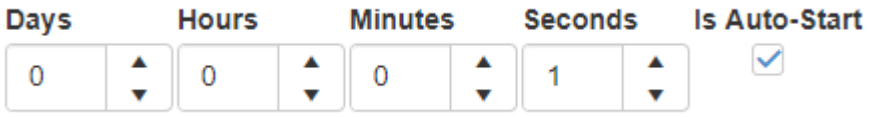
To get the *Exact* format to work like the *Comma Delimited* you would have to manually add commas and control characters, turning the above definition into:

Component Type	Value	
Custom Text	$DEMO_CM	Edit Delete
Custom Text	,	Edit Delete
Date / Time	SCS Date & Time - Short	Edit Delete
Custom Text	,	Edit Delete
Sensor	Furuno-GP170_Latitude	Edit Delete
Custom Text	,	Edit Delete
Sensor	Furuno-GP170_Longitude	Edit Delete
Custom Text	,	Edit Delete
Sensor	Pressure	Edit Delete
Control Character	CARRIAGE RETURN	Edit Delete
Control Character	LINE FEED	Edit Delete

While the end result of the above examples are the same, the *Exact* format gives you much finer control. However, it is also much more verbose, harder to read and harder to maintain. For that reason it is recommended you only use the *Exact* format as a last resort.

Update Rate

The *Update Rate* allows you to specify how often you want your message sent out. It also has an option to have this message automatically started whenever the service is started (e.g. after the entire server is rebooted). In the example below the message is sent out once every second and will automatically start sending whenever the service comes online.



> Custom Messages can also be automatically started/stopped via GIS Triggers.

Output Configuration

The Output Configuration section is where you define where you want your messages sent. Depending on which output type you select you will be presented with different options to further refine your target.

Serial Port

This output will push your data out a serial communications port located on the server itself. The settings you specify here will have to be matched to whatever is listening in on the other side of the serial line.

TCP/IP

This output will push your data out from the server via TCP. This is a good reliable communications protocol over a network, however the computer the client software / target is running on will have to be able to connect to the SCS server over the network. This may require you to open firewall or ACL holes in your network(s) depending on which VLAN the various systems are running in. In a client-server architecture SCS acts as the server in this instance and remote clients should connect directly to it on the port specified.

UDP/IP

This output will push your data out from the server via UDP. This is another protocol which sits on IP, it is not reliable but excels at broad announcements (broadcasts) and speed since it doesn't have all the overhead associated with TCP. However, as with TCP, it is a network based communications protocol and the computer the client software / target is running on will have to be able to connect to the SCS server over the network (or packets from the SCS server have to be able to reach the client machine). This may require you to open firewall or ACL holes in your network(s) depending on which VLAN the various systems are running in. Multi-cast might require further switch level modifications, consult with your network administrator for more information. In a client-server architecture SCS acts as the server in this instance and remote clients should listen for packets from it on the port specified.

SignalR

This output will push your data out from the server server via a SignalR hub. This is the same technology SCS uses throughout the website for most near real-time communication between all users and the backend services. There are various libraries written in python and other languages if users want to directly connect via SignalR and use it in their own applications.

> The name of the Hub to connect to is "*UserDefinedMessage*"

File

This output will push your data out to a file directly on the filesystem.

> ⚠ If trying to write to a network drive you might want to use a FQDN and ensure the share has granted permissions to the account the Data Dissemination Service is running as. Be aware drive mappings are tied to a profile and have proven to be somewhat flaky in the past.

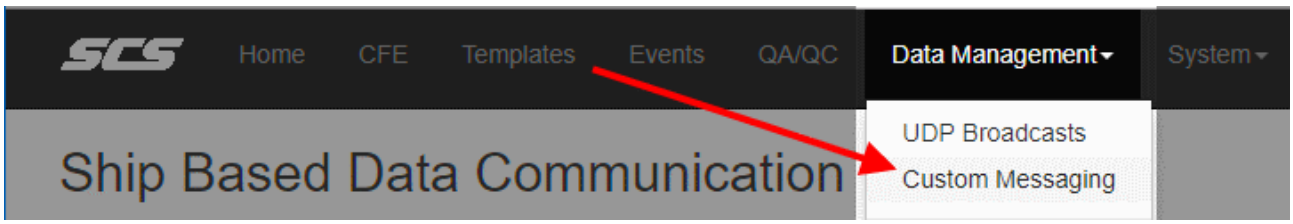
Email

This output will send your data out via an email at every elapse of the Update Rate interval.

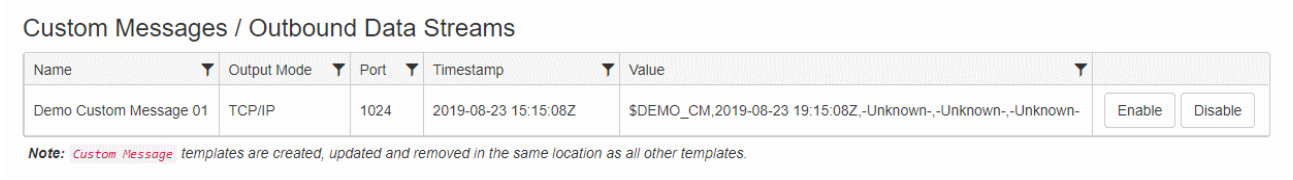
> ⚠ Having a low *update rate* could result in saturation of your email system. Be careful using this if you have a low *update rate* (e.g. if your *update rate* is 1 second then every second an email will go out with the message, which is 86,400 emails per day just for the single `Custom Message`!

Run Time

`Custom Messages` are run via the *Custom Messaging* page inside the *Data Management* main menu item.



Each template is displayed in a grid, the last column allows you to start or stop the message from going out.

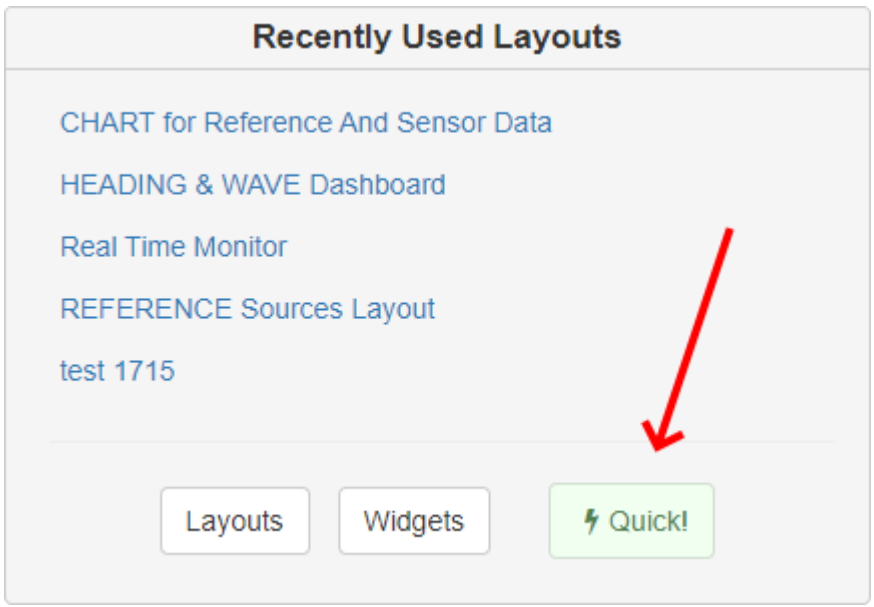


Quick Visualization

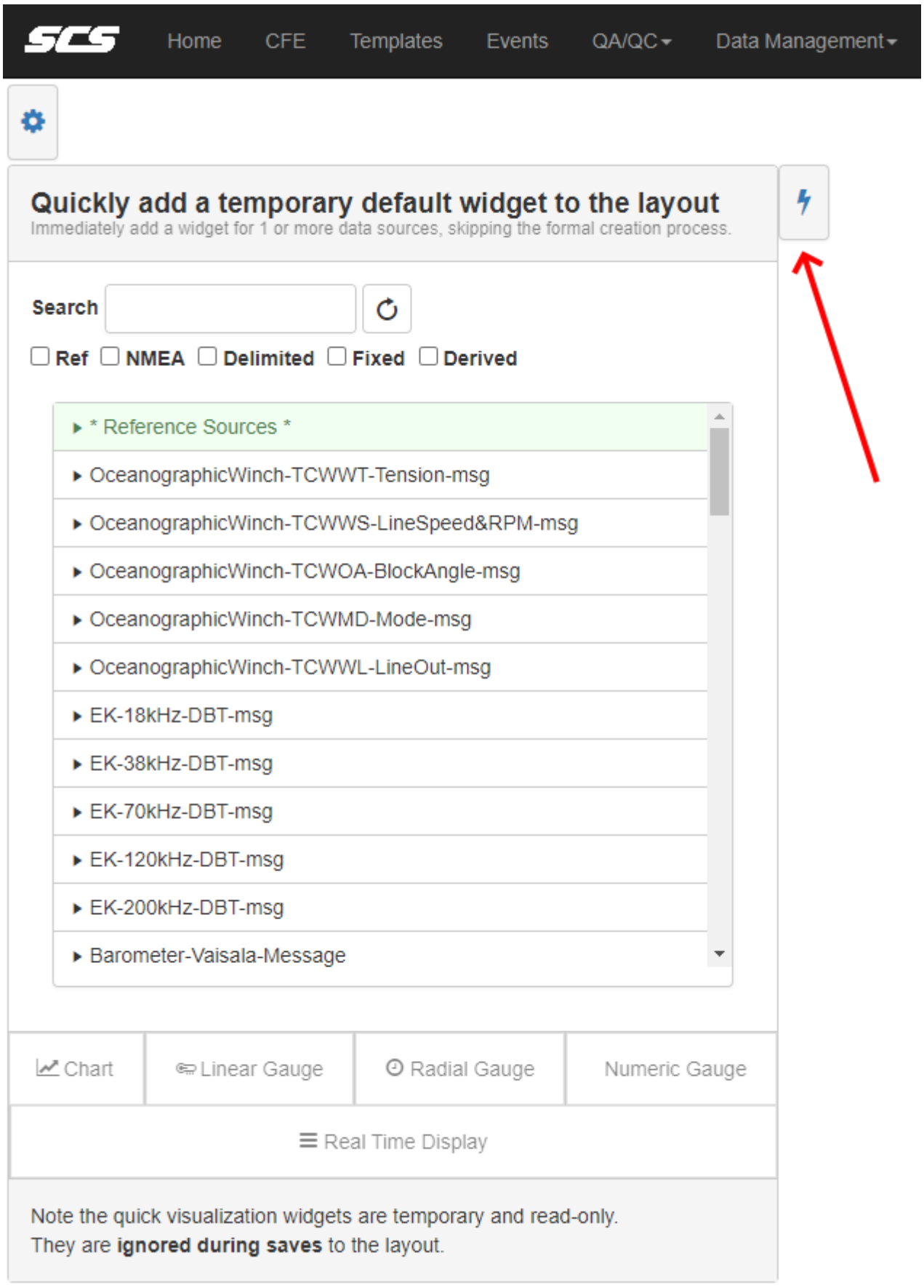
Widgets are the main data visualization tool for users and sit at the lowest level of the user defined interface. However, sometimes it is inconvenient to create a dedicated template to view a stream of data from a DataField. For instance, let's say you have a layout already in place and you want to quickly view data from a single sensor. The normal procedure would be to create a new Widget dedicated to this feed, add it to a Widget Group and add the Widget Group to the Layout. Aside from a lot of steps this also creates permanent templates saved into your database.

An alternative workflow could be utilized via the ***Quick Visualization*** slide-out. This feature allows you to instantly add a data stream to your layout presented in the Widget of your choice. This Widget is ephemeral and as soon as you leave the page or close the Layout it disappears. It is not saved to your database and cannot be customized.

To add a *quick visualization* Widget simply click the *Quick* button on the main homepage



or click the lightning slide-out handle in an active layout.



To add a widget simply select the source(s) you want to visualize and then click the button at the bottom for the method you want it displayed.

⚠ Quick Viz widgets cannot be modified or saved. If you want a tailored widget please go through the normal procedures.

Introduction

SCS offers multiple styles of data logging. The traditional style is continuous ACQ logging, which records complete raw sensor data into multiple types of outputs (RAW files, NetCDF files, database tables, etc). Each sensor is logged to its own file, and in post processing it is necessary to correlate the contents of any two sensor log files using timestamps.

The second style is Event Logging, which provides a means of collecting data sets that are completely separate from the raw data collected by the system. In a typical situation, you may only be interested in sensor data collected only during specific times within the cruise. Typically these times represent specific events that may take place during the cruise, such as a CTD cast, a trawl tow, a balloon launch, a buoy deployment, etc. Even Logging enables each and every scientist or end-user to collect a customized set of data during a cruise. At the end of the cruise, the user will walk away with exactly the data he/she desires, without the need to sort through the entire raw data set.

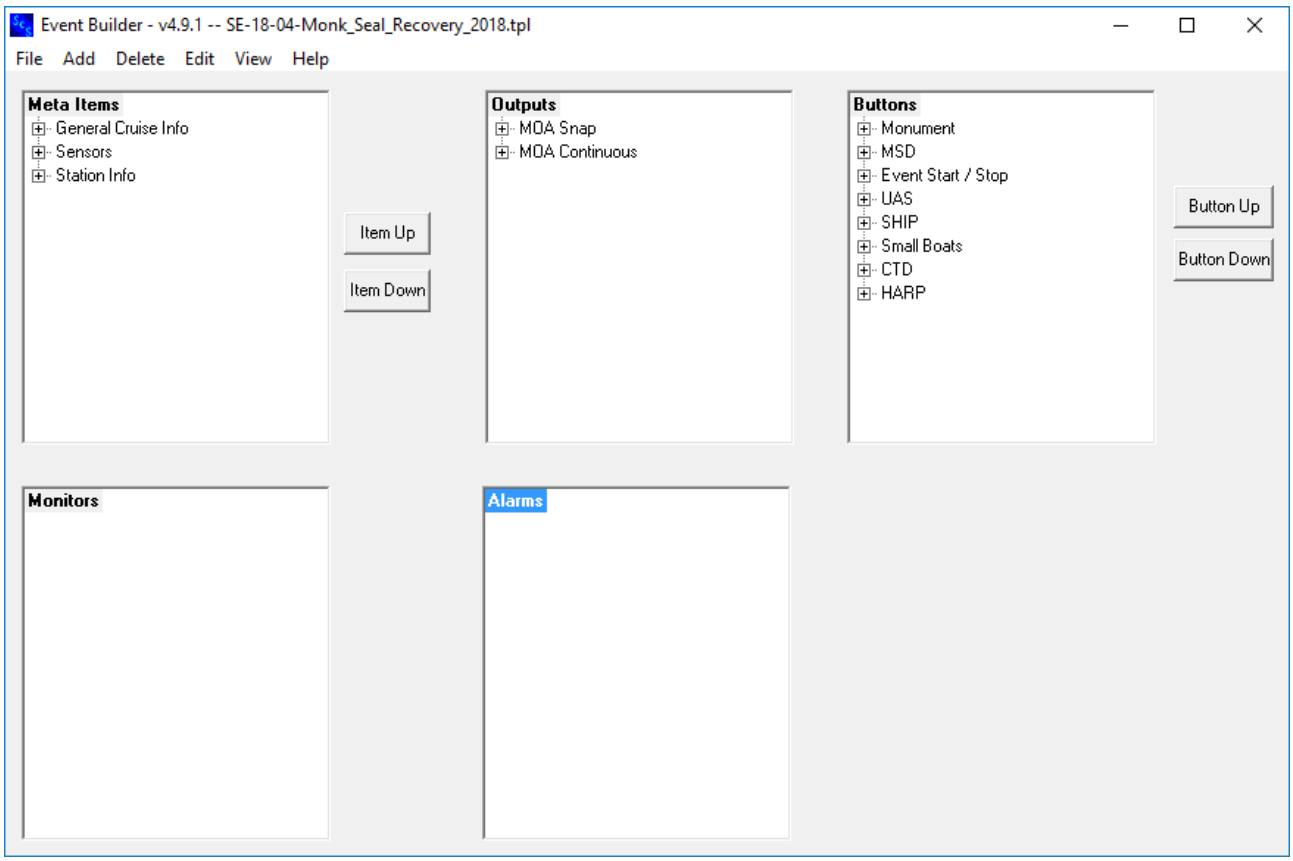
Event Logging has two main interfaces on the SCS site. Similar to other widgets you must first use the Event Template Editor to define an event. Once you have a template it can be executed via the Event Management page. On the back-end all the work is being done by a windows service (SCS Event Logger). The Logger in turn makes use of the website's SignalR data hub to access any sensor data it may need.

In this documentation, event refers to a particular operation on the ship, such as a fish trawl or a CTD cast. SCS Event Logging allows the user to define specific actions within an event, such as "CTD in water", "Net deployed." For any action you can choose to record the current value of sensor data, manually entered data, or clock time. When the event is run, each action in the event is initiated by the user by clicking on buttons in the Logger GUI or via user defined automated triggers.

Legacy Event Logger

Events in SCS v5.1.0 were designed to try to incorporate the ideas and terminologies used in the older versions of SCS in order to ease your transition and reduce the required learning curve. Many features you may be used to such as Meta Items and Outputs have been reproduced in v5.1.0 though they may be tweaked in a number of ways. Many features of OEL have been brought into the fold as well as many requests from users like you. However, the new framework does support exporting the results of an event into the same legacy file setup (HDR, ELGs, etc) to ensure backwards compatibility with existing applications.

See Also: Event Logging SCS v4 to v5



Event Logging - SCS v4.x to v5.1.0

Help for those migrating from SCSv4 Events to the modern framework.

The main purpose of the modern SCS Event Logger remains the same as the prior versions. To that end many of the things you have learned to love (or hate) have been carried over into the new system. After consultation with users we have attempted to add features which have been missing while trying to keep the user interface as similar and streamlined as possible, fixing issues and making areas more intuitive along the way.

Building an Event Template

Meta Items

- Tab Groups have been removed, instead a new UI builder is available to allow you to structure your Event screen as you see fit.
- Manual, Sensor and Summary meta items all have migrated over into the new framework.
  - "List Of Values" is now called "Selection Options"
  - "Date & Time" has been pulled out into it's own type
  - "Beaufort Scale" has been removed

Outputs

- Output to a file/device/socket has been removed

Buttons

- Buttons have been re-termed as Manual Triggers
- There are no explicit start or stop buttons
  - The event starts automatically when launched and is stopped via a special Action
- Button groups are no longer limited to 2 buttons.
- Buttons no longer directly manipulate Meta Items or Outputs, instead they start Action Sequences.

Monitors and Alarms were not actually implemented in v4 however they are in v5.1.0 via the Auto Trigger and "Alarm" action respectively.

The dependency report is now part of the "Validation" tab.

Many of the features from OEL have been included in the new system such as:

- The ability to specify colors
- The ability to group multiple manual triggers (buttons)
- The ability to set a forced press order, allow multiple presses, allow for repeating sequences for manual triggers

Running an Event Template

Event Templates are managed via a website now. The new framework allows multiple users all over the ship to simultaneously participate in a single event, no more 'local machine only' approach. You can join, leave and return to an event without having to start it over, even if the server reboots the event should pick up where it left off. Event data is saved into the database and must be exported / downloaded by a user before it can be worked on. Note anyone with appropriate rights can download the results of an event, it is no longer required to have mapped drives or for the ship [ET/ST] to put everything together to provide to the PI as they can now do it themselves on demand.

Action Sequences

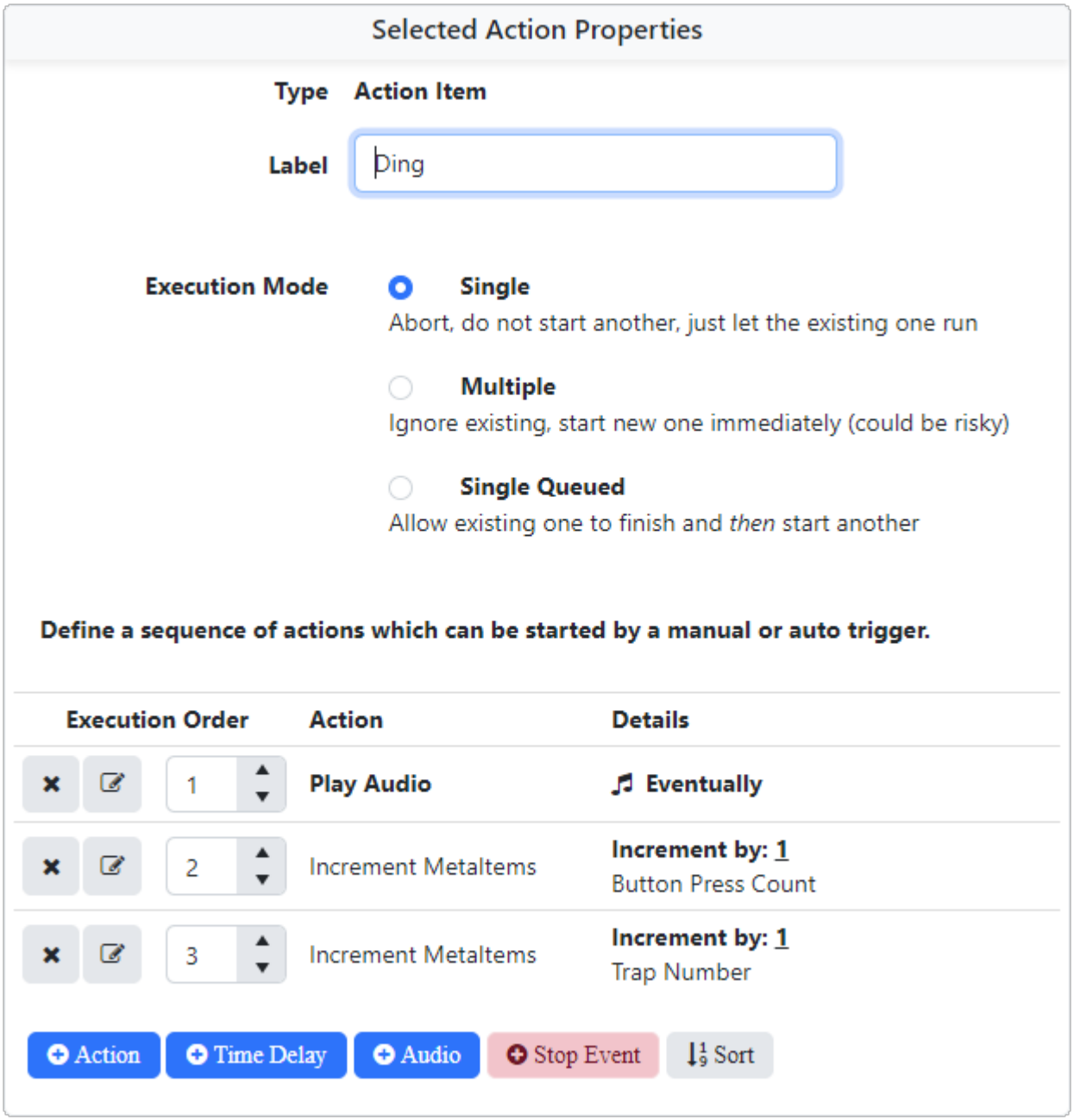
Action Sequences are core to almost all interactions between the various components in an Event. An Action Sequence defines a list of things to do in response to a trigger. The trigger could be manual (Manual Trigger / button) or automated (Auto Trigger). Action Sequences are the meat of the Event and are arguably the most complicated part. All user created Manual Triggers and Auto Triggers have to start at least one Action Sequence. The Action Sequence abstracts the actions which should be performed in response to the trigger.

Action Sequences can operate in 1 of 3 modes:

Single
: Ensures 1 and only 1 run of the sequence is occurring at any given moment. If the Action Sequence is triggered again before a prior run of it's action list has completed then the new request is ignored.

Multiple
: Allows runs to be executed in parallel. When triggered the sequence is executed immediately regardless of whether other runs are still being processed or not. This can get very complicated quickly, if you choose this mode do so after thinking through the potential pitfalls.

Single Queued
: Similar to *Single*, this ensures only 1 run of the action sequence is executing at any given moment. However if the sequence is triggered while a previous run is still being processed instead of ignoring it (*Single* mode) or executing it immediately (*Multiple* mode) the *Queued* mode will add the request to a queue. Once the prior run finishes the next one in the queue is executed and so forth until the queue is emptied.



Action Sequences have one of the most detailed *Property Panes* of all components.

The *Execution Mode* is detailed above, depending on the use case you are building your event to accomplish one mode will most likely make more sense then another.

The main part of the Action Sequence is defining the list of actions which comprise the sequence. This is done by using the buttons at the bottom of the *Property Pane*.

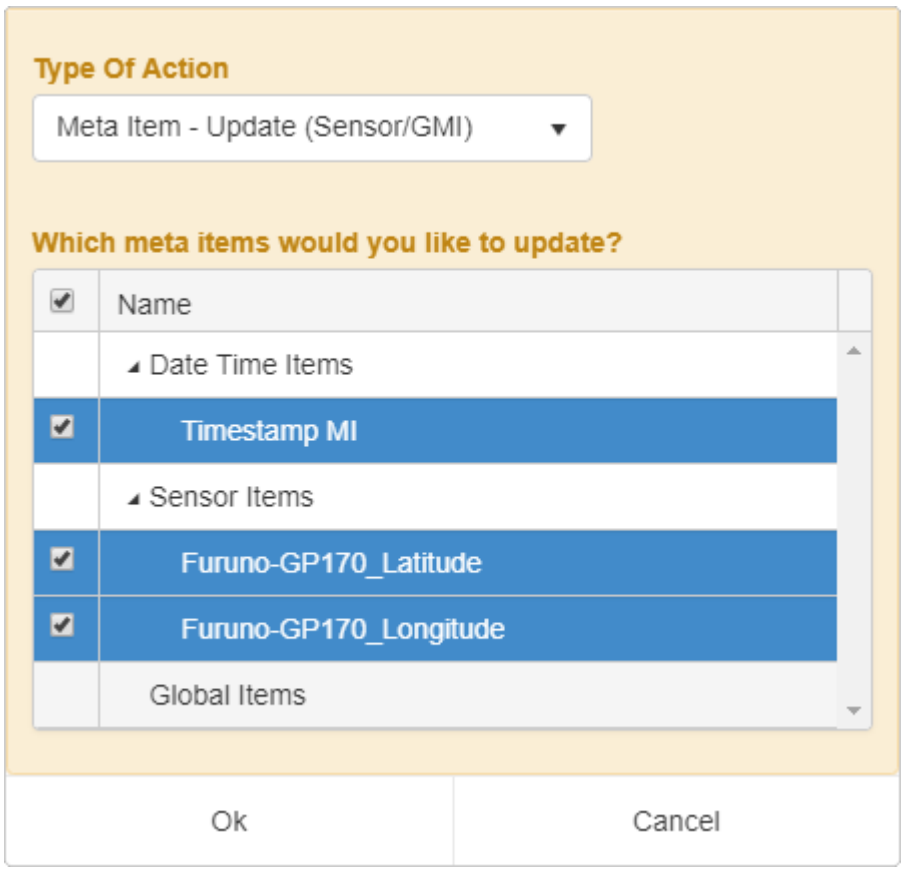
Each action item added to the list is executed sequentially based upon their assigned *Execution Order*. If two actions have equal *Execution Orders* then they are executed in parallel.

The *+Action* button allows you to add an action item to the sequence. Clicking it will pop up a dialog with all the various action types you can execute.

> Some values might be empty or not make sense until you create a required component. For instance you cannot add an action to start an Output until you have an Output defined.

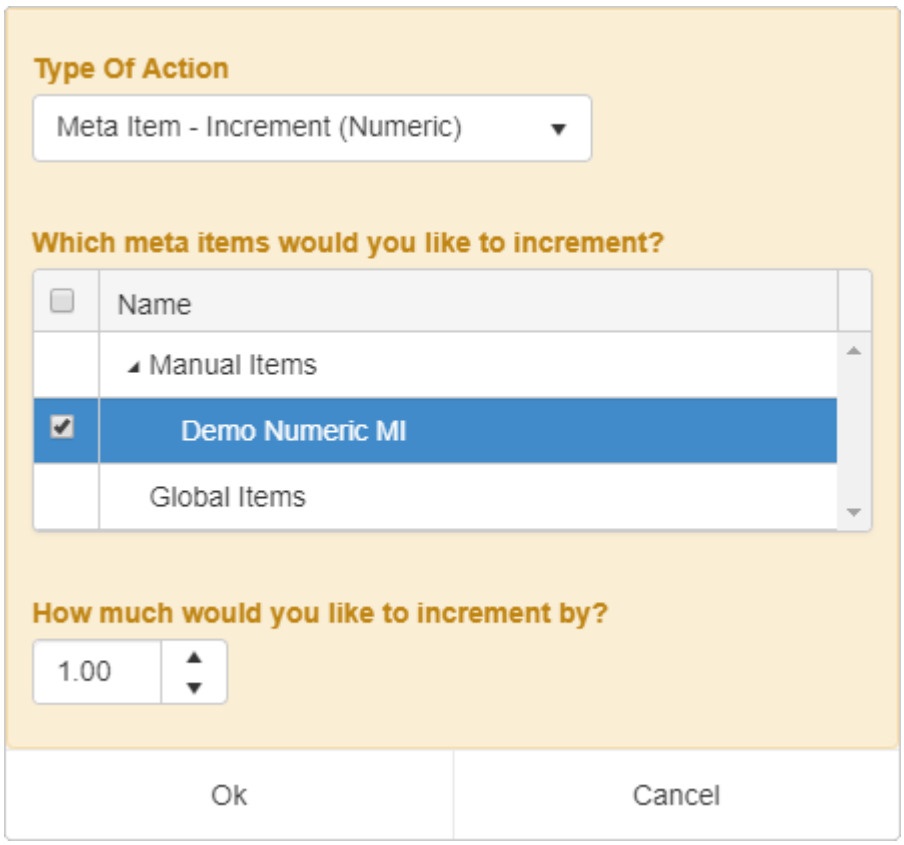
Meta Item - Update

This action will update any sensor or global Meta Item components you check. When executed Sensor Meta Items will be updated to the latest value obtained from the Data Hub, Global Meta Items will get their current value from the Event Logger Service.



Meta Item - Increment

This action will increment any numeric Meta Item you check. You can define the amount you wish to increment by. Decimal increments will be ignored if you try to increment a numeric Meta Item of type Integer.



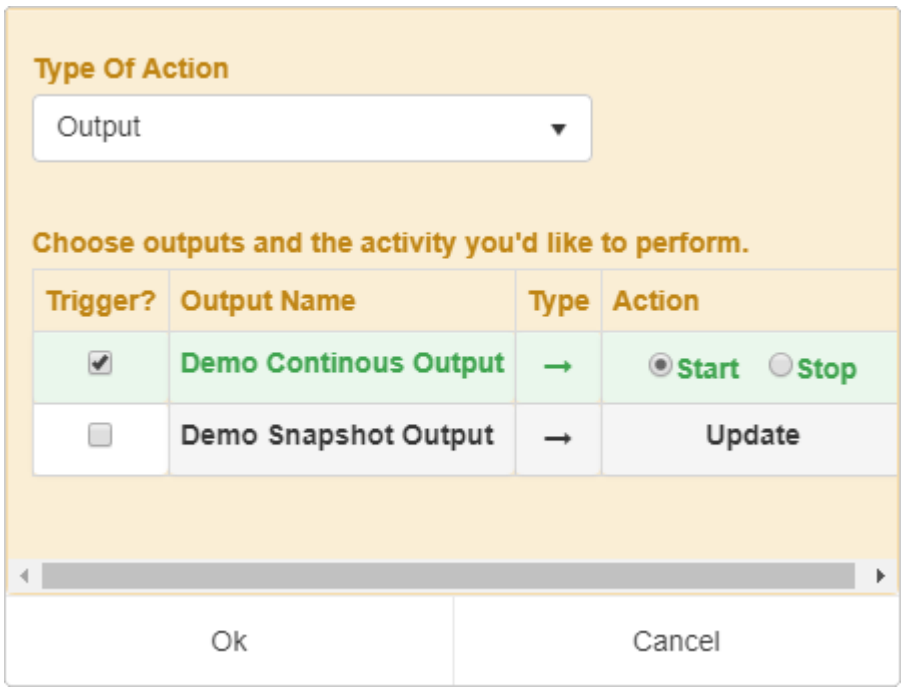
Meta Item - Change

This action allows you to override whatever the current value of the checked Meta Items are with another value. This will only work for Manual Meta Items, you can not change sensor, global or summary Meta Items.

This can be a useful way to provide a lot of functionality in your UI without taking up much space. For instance, lets say you have 20 types of objects you want to deploy off the ship. Lets say you want to deploy 10 of one, then proceed to deploy 10 of the next type, etc. You could have 20 Meta Items, each representing the number of object 'x' currently deployed. Or you could have a single Meta Item and once it reaches 10 you simply reset it's value back to 0.

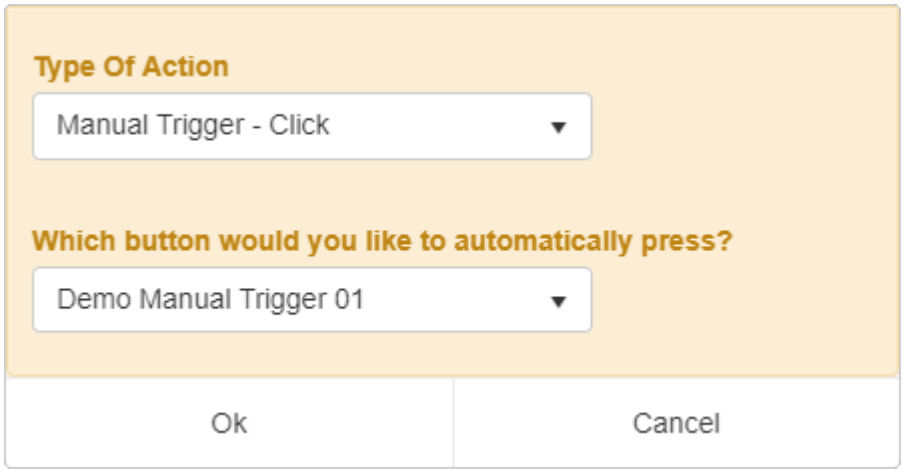
Output

This action allows you to perform actions on the various Outputs you have defined. If the Output is *Continuous* you can start or stop it. If it is *Discrete/Snapshot* then you can update it. Simply check the Output you want to act on and select the action you wish to perform if applicable.



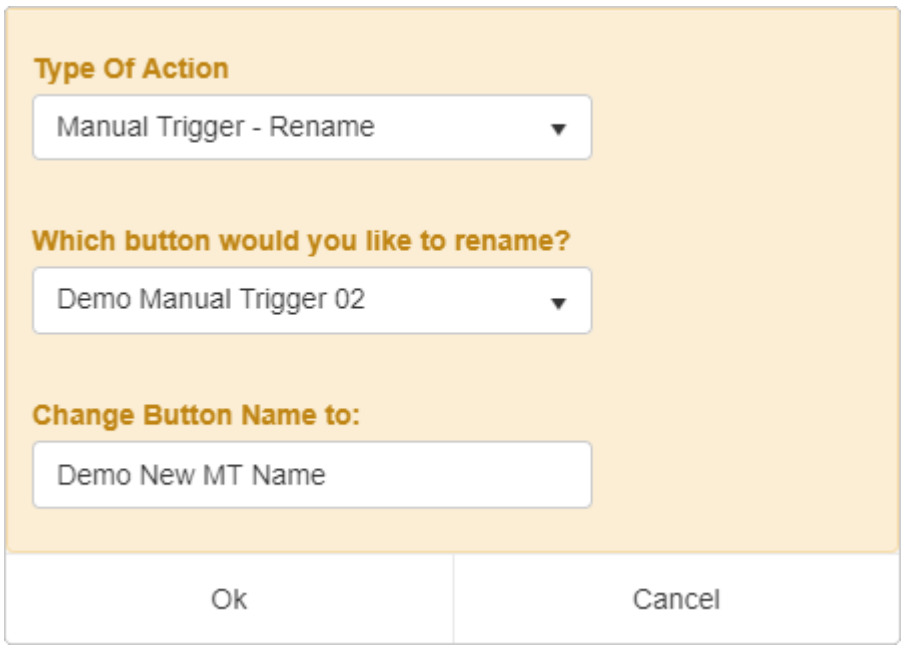
Manual Trigger - Click

This action allows you to automatically press a Manual Trigger. This will cause the system to respond as if you manually clicked the button and will trigger all Action Sequences associated with that trigger.



Manual Trigger - Rename

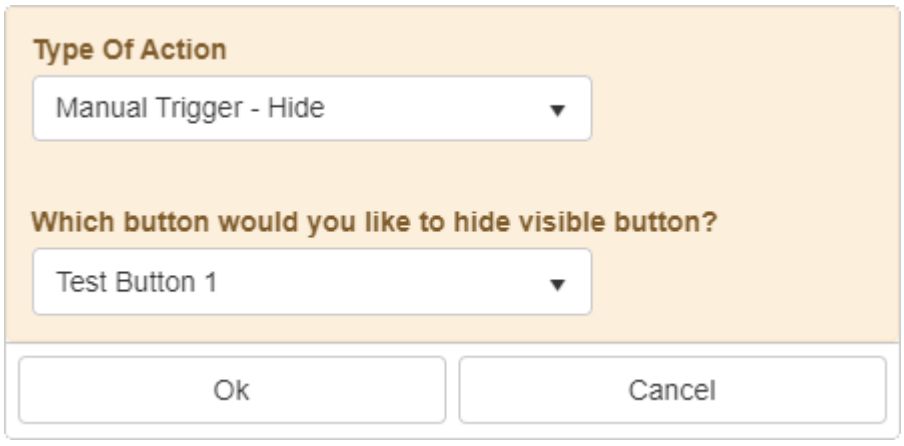
This action is similar to the *Meta Item - Change* action mentioned above, however this action acts against a Manual Trigger. Essentially this allows you to automatically change the *Label* of an existing Manual Trigger, so for example if you had 2 types of buoys to deploy and you wanted to deploy 10 of Buoy A and then 10 of Buoy B then you could create a Manual Trigger named "Deploy Buoy A" and after 10 presses simply change it's name to "Deploy Buoy B".



Manual Trigger - Hide

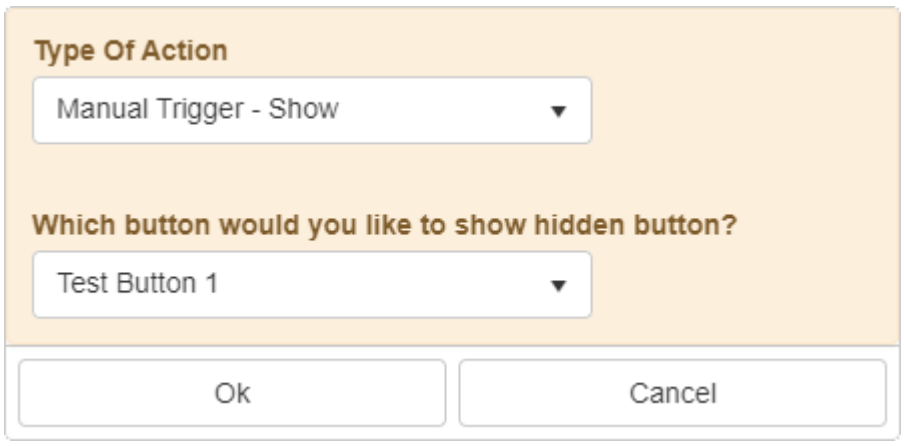
This action removes a Manual Trigger from the screen, thereby preventing users from clicking on it as they cannot see it to begin with! This works in conjunction with **Manual Trigger - Show** by removing a visible button from the UI. It has no effect on buttons already hidden.

> ⚠ Be careful hiding buttons which belong to groups as you *may* put your event into a logically unsound state! For instance, if you have a group with forced order and you hide a button then it cannot be clicked and you may block that group from proceeding until the button is shown. This may be the exact logic you wish to implement of course, but just be sure to give these workflows thought when hiding/showing.



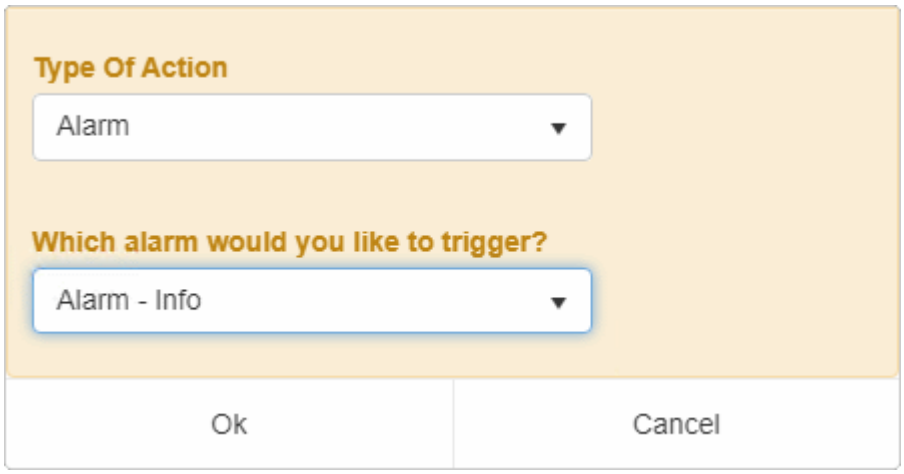
Manual Trigger - Show

This action re-displays a Manual Trigger from the screen. This works in conjunction with Manual Trigger - Hide by putting a previously hidden button back onto the UI. It has no effect on buttons already visible.



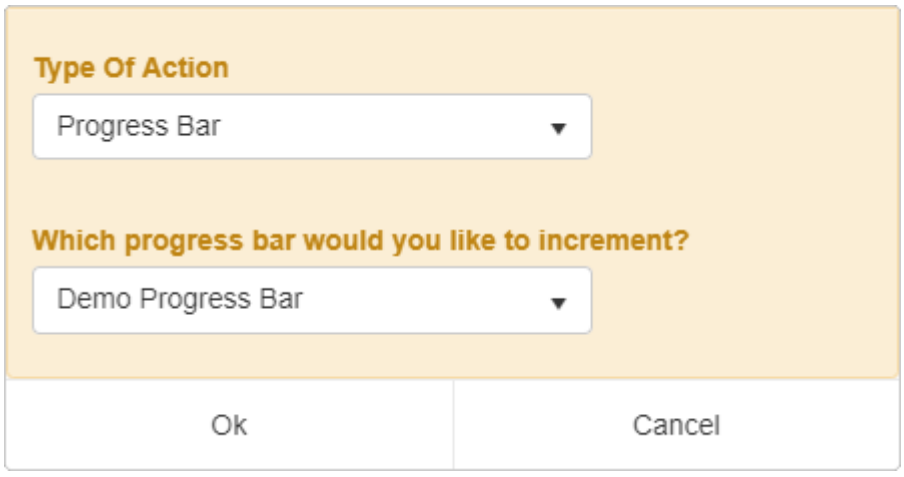
Alarms

This action is used to trigger alarms defined inside this event template.



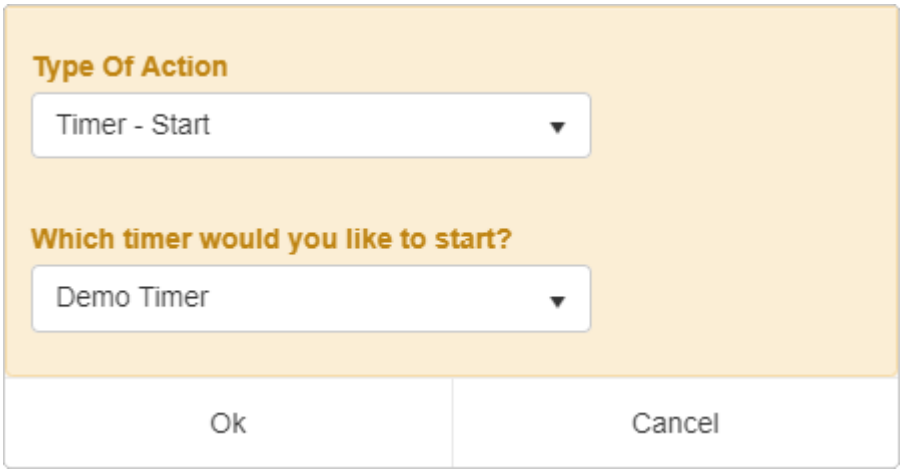
Progress Bar

This action is used to increment a progress bar by 1.



Timer Start / Stop / Reset

The timer actions act upon existing *Timer* components in the exact manner in which they are described. Use the *Timer Start* action to start a timer, use the *Timer Stop* action to stop a timer and use the *Timer Reset* action to reset a timer back to 00:00:00.

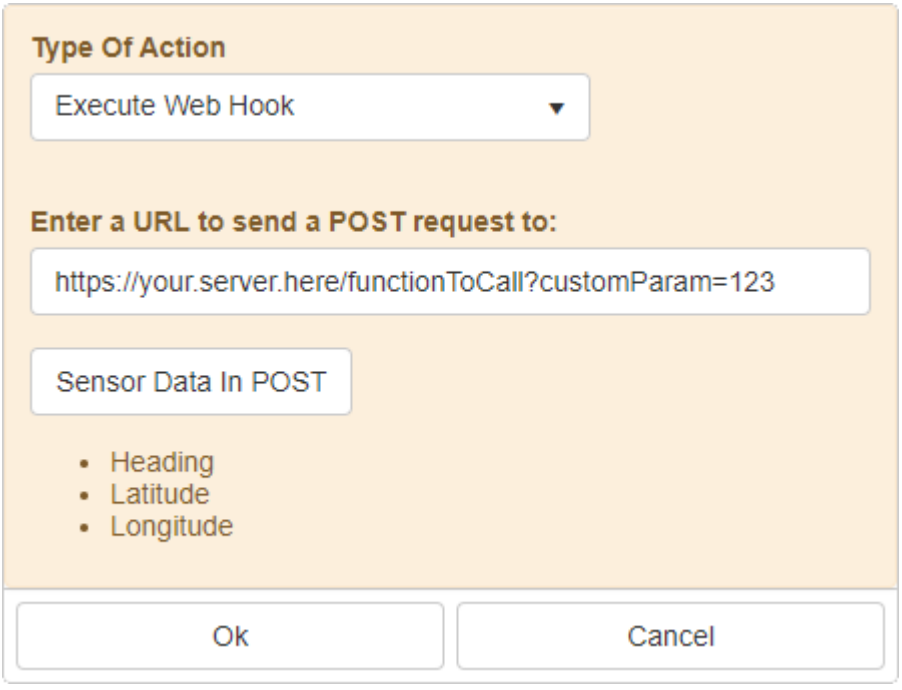


Execute Web Hook

This action will make a call (POST) to a remote RESTful API. This can be used to trigger procedures in other systems external to SCS, for instance launch some custom python code written by the science party or update another systems database or website, etc. This requires a remote web-service to be running and accessible over TCP/IP from the SCS server.

You can include URL parameters as you deem necessary however these will be static and part of the hard-coded URL in the action.

SCS sensor data can also be sent in the call if needed. If you select data field values to include in the POST then SCS will add the latest values *at the time the call is made* to the body of the request.



Details on handling this in your code can be found in the Event API Action section of the documentation.

The ***+Time Delay*** button allows you to add a time delay to the sequence of actions. Clicking it will pop up a dialog where you can enter the amount of time you wish to delay. Any action items which are in the next higher execution order will be blocked and will not start until this timer completes. If no additional action items are in the sequence then the sequence itself will continue to block other instances (if execution mode is not *Multiple*) until the timer completes.

A *Time Delay* can be used whenever you know beforehand the static amount of time needed between two actions. For instance, if you know that a trawl net should only be on bottom for 30 minutes (randomly chosen duration for demo purposes only) then you can setup a timer to start when the net hits bottom and then send out alarms or perform any other actions automatically after 30 minutes.

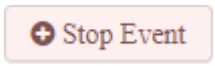
> ⚠ Be careful with the *Execution Order* here, setting the wrong order can render your timer pointless.

The ***+Audio*** button allows you to insert audible sounds into the sequence of actions. Clicking it will pop up a dialog where you can select a short sound clip to play. As you select different options you'll hear them play locally. Click the green arrow/play button to re-play them if desired. Any action items which are in the next higher execution order will be blocked and will not start until the sound clip completes.

Audio files can be linked to multiple elements of the Event via this Action. For example, you could play a sound whenever a user clicks a button. You could play a sound once you've deployed 10 buoys (utilizing an Automatic Trigger with a Meta Item reference). You could play a sound when a timer expires and so on.

> If you're unsure which sound to use just select the first sound clip and use the up/down arrows on your keyboard to move through the list. This will play each sound clip for you without a lot of mouse clicking.

> You can also chain audio clips by taking one short one and repeating it with another *Audio* action over and over again making it a longer one. This would be more useful for an alarm or something you REALLY want people to pay attention to.



The ***+Stop Event*** button allows you to stop the event. In prior versions of SCS this would be the equivalent of manually clicking the "Stop Event" button. By putting this into an action you are able to stop the event via a variety of means. You can still create a "Stop Event" Manual Trigger of course, however there may be cases where one or more triggers would be cause to end the Event. For instance perhaps in a trawl event once the net is on-board you would normally want to end the event. If so when creating the Action Sequence to handle the net coming on board simply add a *Stop Event* action to the end of the sequence. When the net comes aboard the event will automatically end.

The ***Sort*** button automatically sorts all actions in the list according to their currently assigned execution order. It does not change the execution order, only sorts the list so you can visualize the sequence in the order things would execute. This would only be relevant if you have changed the order of one or more action items, otherwise actions are added in a sequential manner by default.

List of Actions

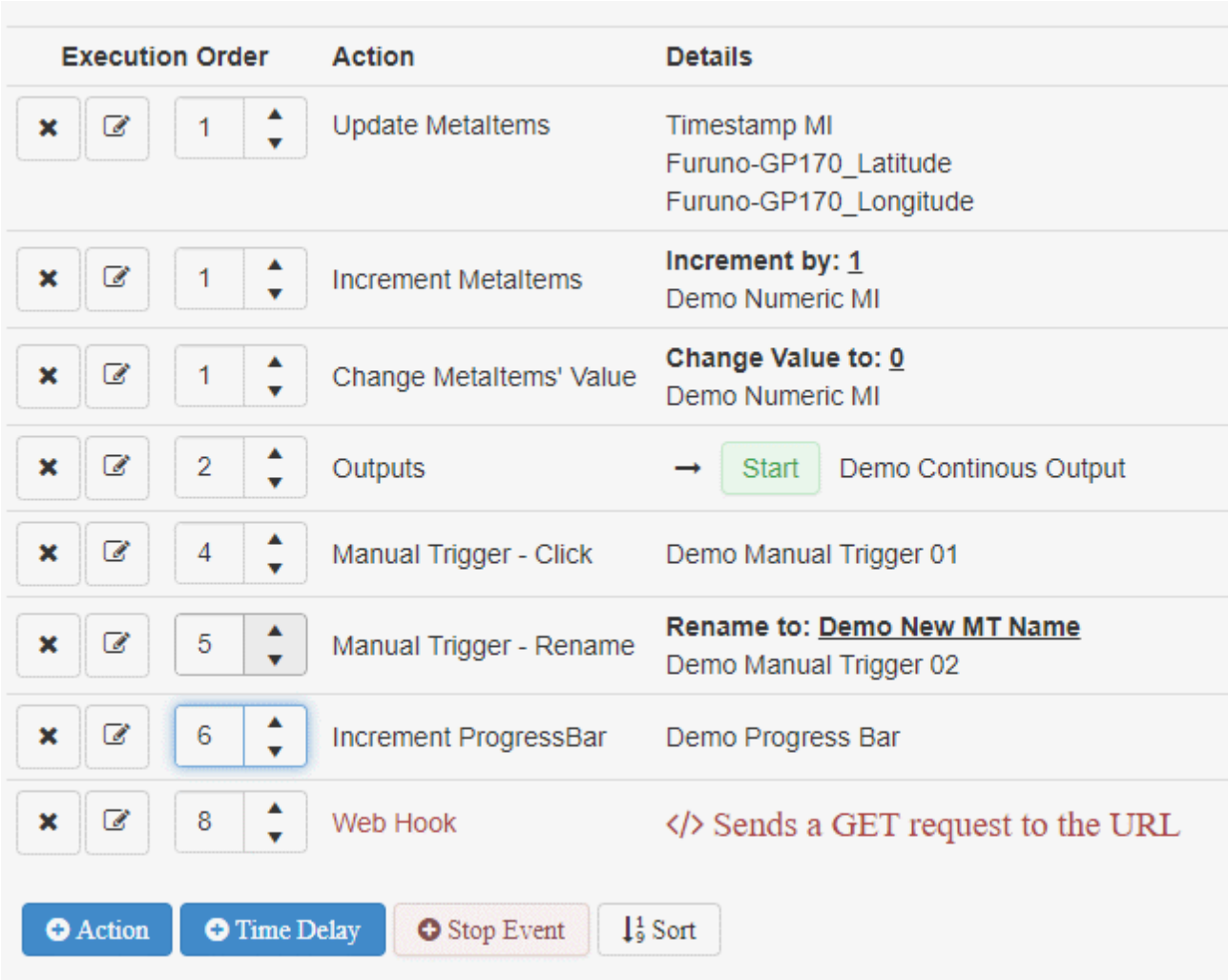
As you add actions to the sequence they will be displayed at the bottom of the *Property Pane*. The first column provides a button to *Delete* the action from the sequence, followed by a button to *Edit* the action. The next column allows you to change the actions *Execution Order* which is relative to other actions.

For instance, in the example below there are 8 action items to be executed when this sequence is triggered. Since the first 3 have the lowest *Execution Order* they will execute first. Since they have the same *Execution Order* (1) those three actions will execute in parallel.

Once all three of those actions are complete the next action in the order will execute (2 - Start the Demo Continuous Output).

Once (2) has finished the next in the list will execute. There is no action with a *Execution Order* of 3 so the next in the list would be (4 - Manual Trigger Click).

This continues until we get to the last action (8 - Web Hook), once that one completes the sequence is ended.



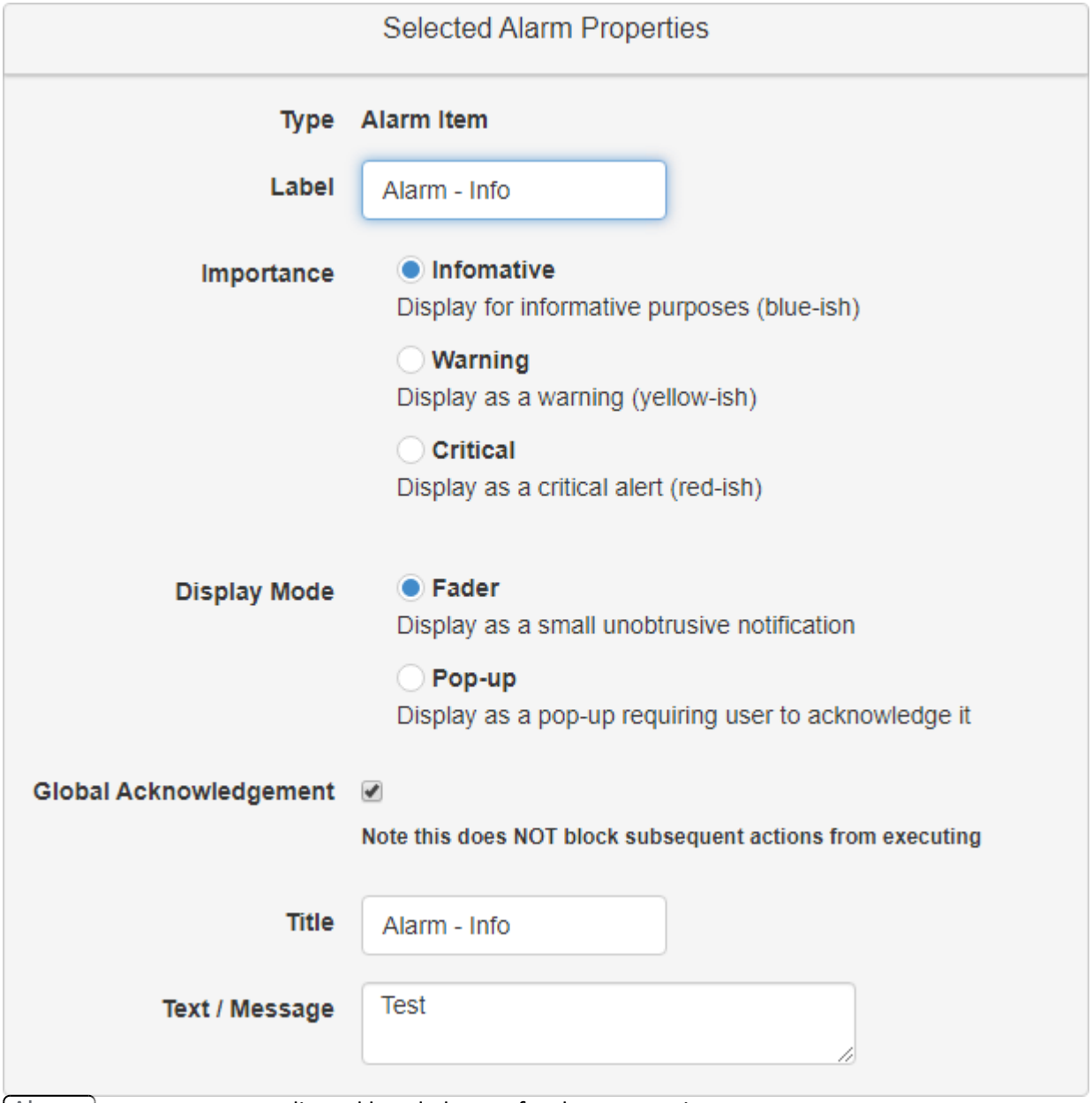
⚠ Be aware that the Execution Order values are what determine the sequence of execution, NOT the visual order in the list. If you change orders it's best practice to hit the ***Sort*** button once you are done so you are not accidentally misunderstanding the order of execution.

Alarms

Alarms are a component in an Event which displays an alert to all users running the event. Similar to Images and Labels, they do not actually interact with or have any impact from a technical perspective on the Event. All Alarms do is provide a means of notifying users with some predefined text via the Event UI. While simple, this can also be very useful when it comes to notifications for those participating in the Event.

Alarms are triggered via Action Sequences similar to many other Event components.

See Also: Action Sequence - Alarms



Alarms are not very complicated but do have a few key properties.

The value you place in the *Label* property is simply for your own reference in the builder and is not displayed during run time.

The *Importance* section allows you to define how you want the alarm visualized when it appears to on the participants screen. There are three choices which are detailed below.

The *Display Mode* allows you to define how interactive you want the alarm to be. If you select *Fader* then the alarm will simply appear as a notification bubble on each users screen and then fade away after a few seconds.

If you choose *Pop-up* then the alarm will display as a pop up box which requires the user to acknowledge it (hit OK) before being able to continue interacting with the event.

The *Global Acknowledgement* setting only applies to alarms you have set to *Pop-up*. It determines if a single acknowledgement is sufficient or if each participant must acknowledge the alarm independently. In other words, when an Alarm is triggered and a popup appears to all the users if the *Global Acknowledgement* option is checked then when the first users clicks "OK" to dismiss the alarm it will

simultaneously disappear from all other users displays. If *Global Acknowledgement* is left unchecked then each participant will have to click the "OK" button independent of each other.

> Note: If *Global Acknowledgement* is checked then users who join the event will be presented with any active Alarms also until someone somewhere clicks OK (it will be persistent).

The *Title* will appear as the title of the Alarm if *Pop-up* is selected.

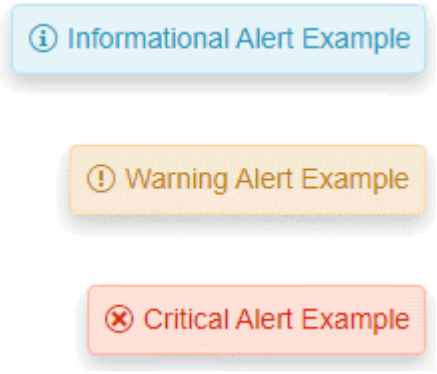
The *Text / Message* is what you wish to convey to the participants. This is the message that will be displayed to all the users.

Importance

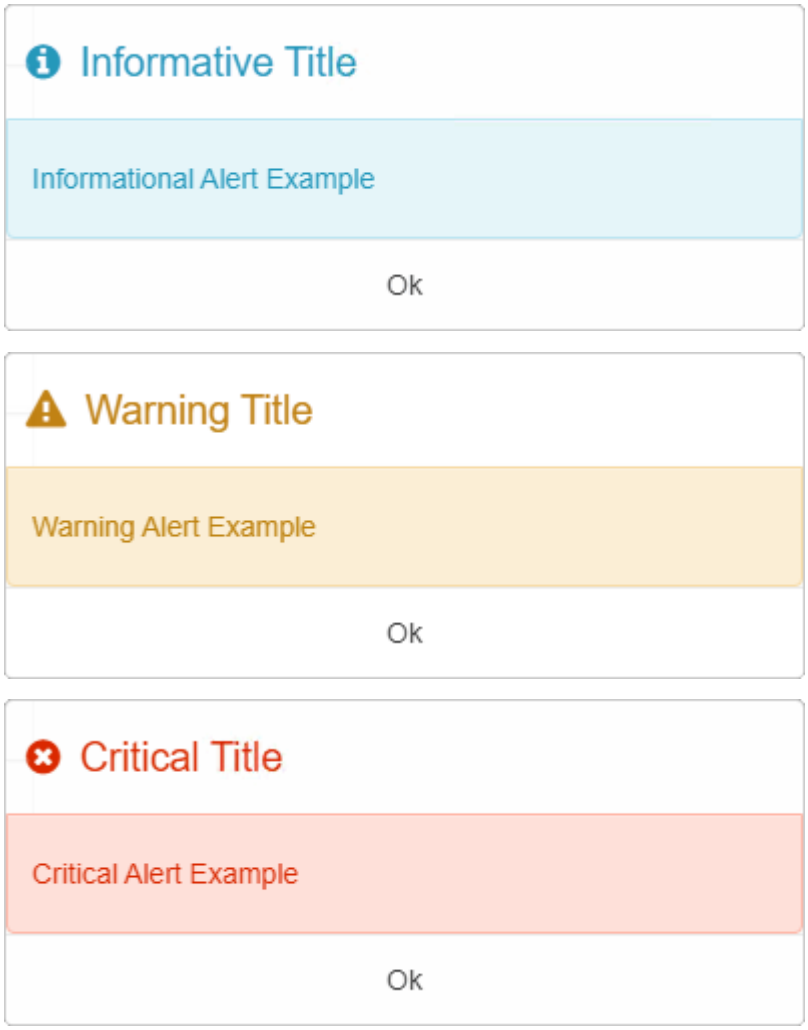
As stated above, the *importance* value determines the color and level of criticality displayed to the user. *Informative* alarms are displayed in a blue-ish format with, *Warning* alarms are displayed in a yellow-ish format and *Critical* alarms are displayed in a red-ish format.

Display Mode

Here are examples of the various *Importance* options displayed as with the *Fader* option set.



Here are examples of the various *Importance* options displayed as with the ***Popup*** option set.



Alarms are visual only. If you wish to include audible alerts with your Alarms be sure to add an Audio action into the triggering sequence!

Automatic Triggers

Automatic Triggers are similar to Manual Triggers (buttons) in that they provide a way to launch one or more Action Sequences. There is one very important difference however, Automatic Triggers do not require a user to become involved, there is no button to click, instead certain conditions have to be met.

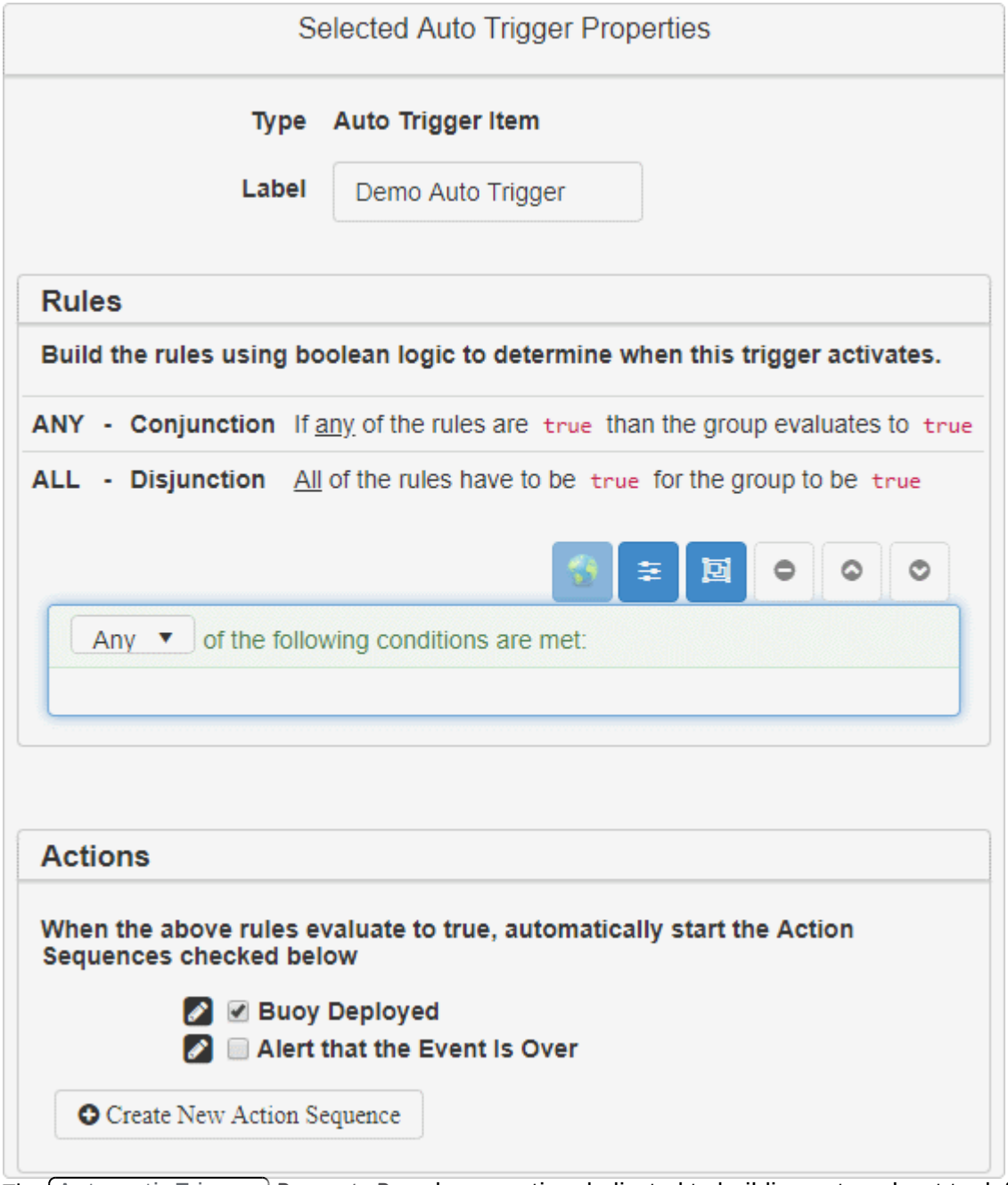
The conditions which have to be met are defined beforehand when you build the template itself. There are two Automatic Triggers which are built into all events, you can optionally assign Actions to launch to these if appropriate but you cannot otherwise modify them.

When the event starts the first Automatic Trigger is fired. If there are any Actions you would like done immediately at the start of an event this is the hook you should tie into.

The second Automatic Trigger is called right after the event is stopped. If there are any post-event Actions you would like to execute this is the hook you should tie into.

> ⚠ The "When Event STOPS" Automatic Trigger is initiated after the *Stop Event* action is executed inside an Action Sequence. It does NOT STOP the event itself, it is only a hook into a cleanup routine which runs post-event.

In addition to the two provided by the system you can create as many user defined Automatic Triggers as you need.



The Automatic Triggers *Property Pane* has a section dedicated to building out a ruleset to define the conditions you want met prior to the trigger firing off it's associated Action Sequences.

These rules utilize boolean logic (AND | OR operators) to allow you to create combinations of conditions which when evaluated with their logical operators must result in *true*

Conditions can be combined into rule groups if you want to apply different logic for sets of conditions. To add a rule group click the *Add New Rule Group* button.

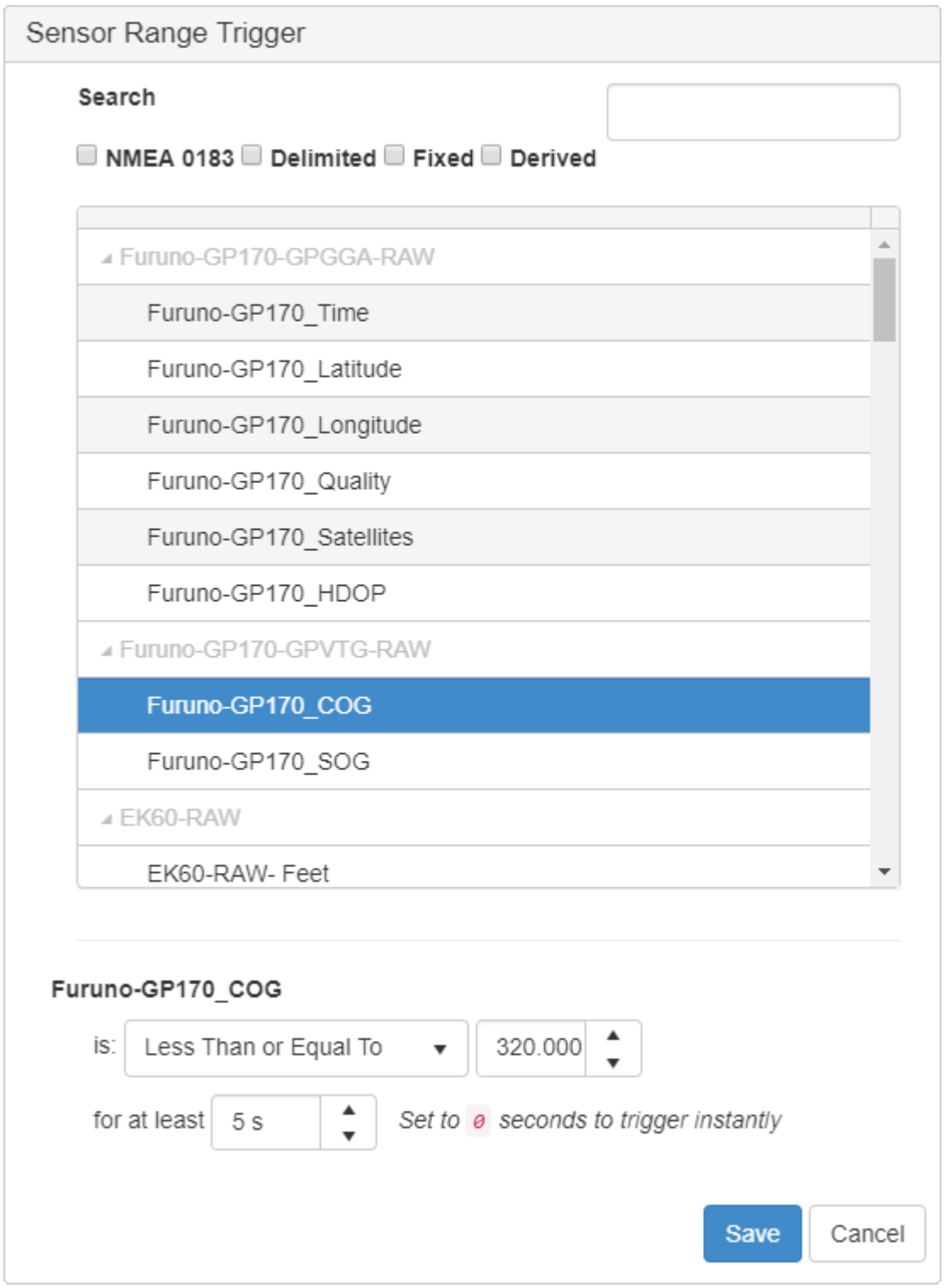
To add a condition first click the rule group you want to add to and then click the appropriate toolbar button (eg the *Sensor Range* button). This will bring up an interface allowing you to define the condition.

As with Manual Triggers, one or more Action Sequences must be checked. These will be what are fired off when the Auto Trigger evaluates to *true*.

> Geospatial rules will be added in a future release of SCS, for now the conditions are constrained to sensor and/or Meta Item values.

Sensor Rules

Sensor rules allow you to specify a condition based upon real-time sensor data.



The first step is to select the Data Field representing the sensor value you are interested in. The top half of the popup will be the standard sensor data source selection control.

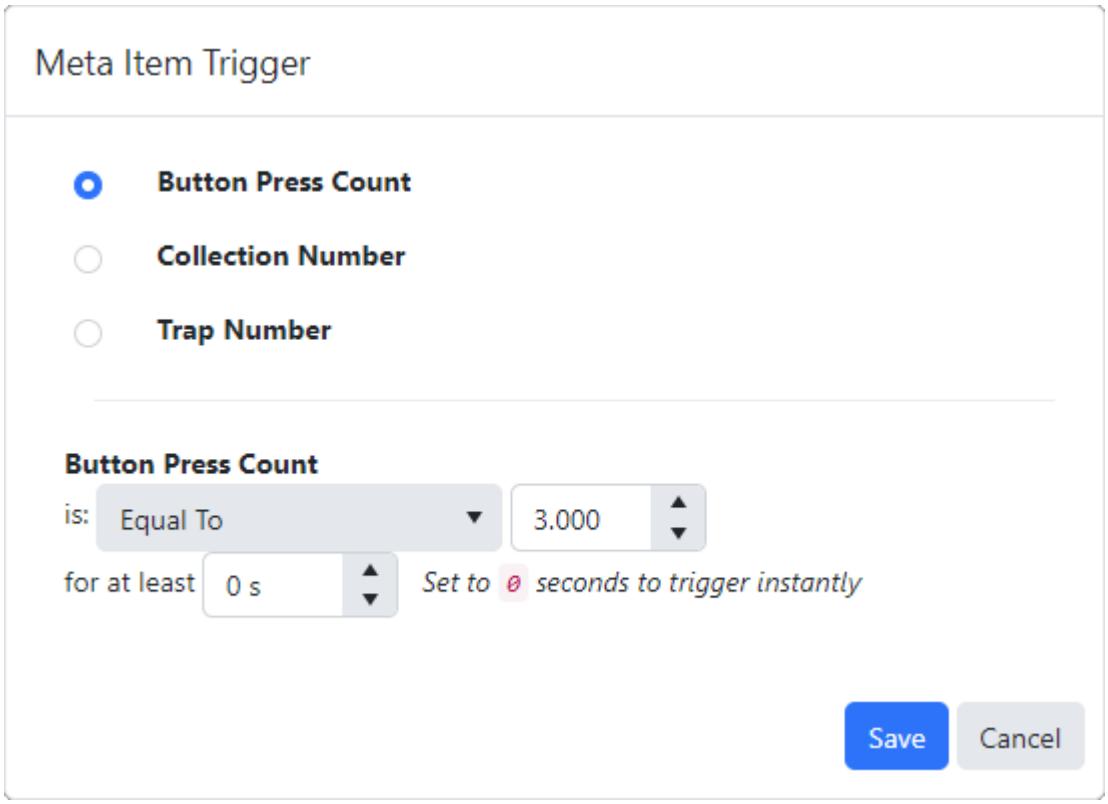
Once you have selected a Data Field you can specify the constraint on its value required for the rule to evaluate to *true*.

In this example, this *Sensor Range Trigger* rule will monitor the Furuno-GP170_COG value. It will evaluate to *false* until the sensor's value is less than or equal to 320 for more than 5 seconds.

> It's recommended to have a little time buffer to ensure the value has actually transitioned. For instance in the above example when the ship turns and it crosses the 320 boundary chances are it will bounce back and forth (319.5, 320, 321, 319 ....). If you did not specify the time qualification then the trigger would also bounce back and forth between true and false.

Meta Item Rules

Meta Item rules allow you to specify a condition based upon real-time Meta Item values.



The first step is to select the Meta Item you are interested in. The top half of the popup will list all known numeric options.

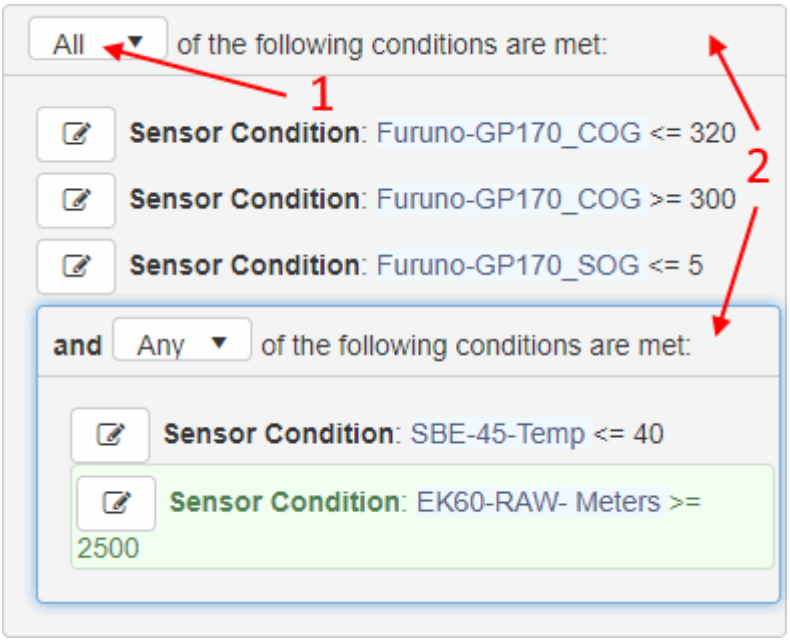
Once you have selected a Meta Item you can specify the constraint on its value required for the rule to evaluate to *true*.

In this example, this *Meta Item Trigger* rule will monitor the Meta Item named 'Button Press Count'. It will evaluate to *false* until the value is equal to 3. As soon as it hits 3 this rule will evaluate to true.

> This rule is very useful if you want to automatically kick off Action sequences at certain thresholds. For example, once you have deployed 10 buoys you could trigger a sequence with an audible alarm. Or, if you want to show/hide a special button every time another button has been pressed 5 times. Or if you have completed 8 transects, and so on.

Rule Groups

Each rule you define has to be placed into a *Rule Group*. When you create a new Auto Trigger a default/root *Rule Group* comes with it. The *Rule Group* evaluates each rule and sub-*Rule Group* it contains and returns *true* or *false* based upon it's logical operator. In the diagram below you can see there are 2 rule groups (arrow #2). Each group has it's own logical operator (arrow #1) and a set of rules inside it which that operator evaluates against as a whole.



The Logical Operator is an ALL vs ANY selection (AND vs OR in the boolean logic world).

All - Everything in the group must be to *true* for the group to be *true.*

Any - If anything in the group evaluates to *true* then the group evaluates to *true*.

So in this example there are 2 *Rule Groups*:

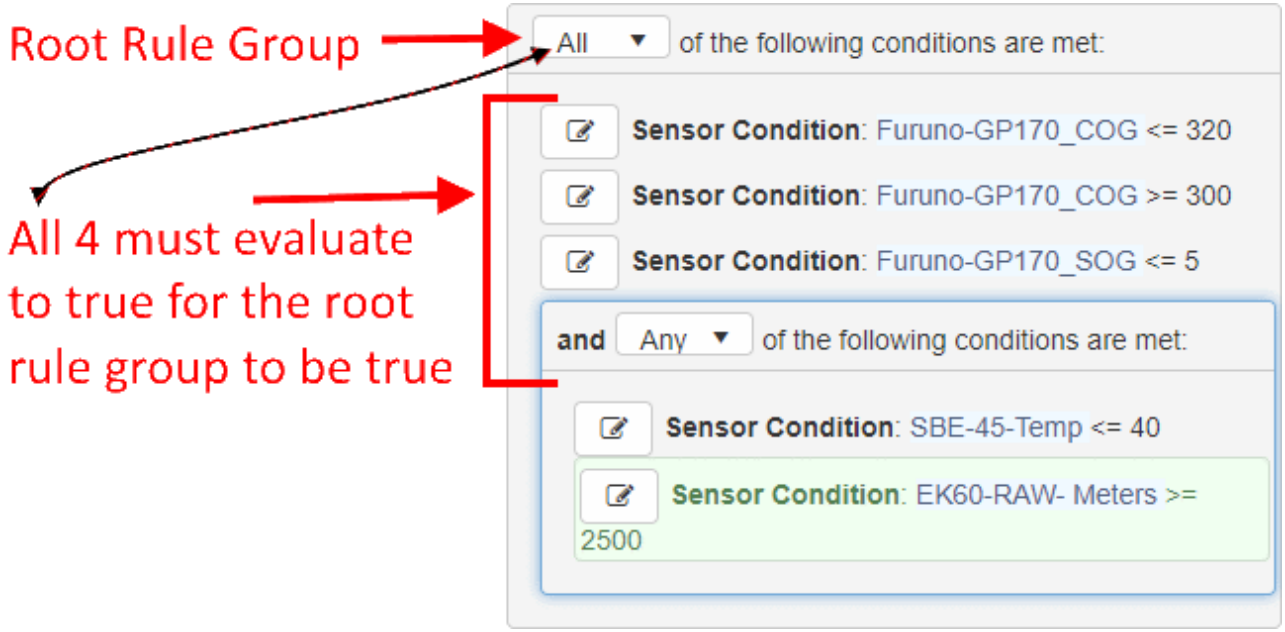
The root one which contains 3 sensor rules and a nested group

A nested one which contains 2 sensor rules

The root *Rule Group* has it's logical operator set to ALL. This means all 3 sensor rules must evaluate to *true* AND the sub rule group must also evaluate to *true* before it evaluates to *true*.

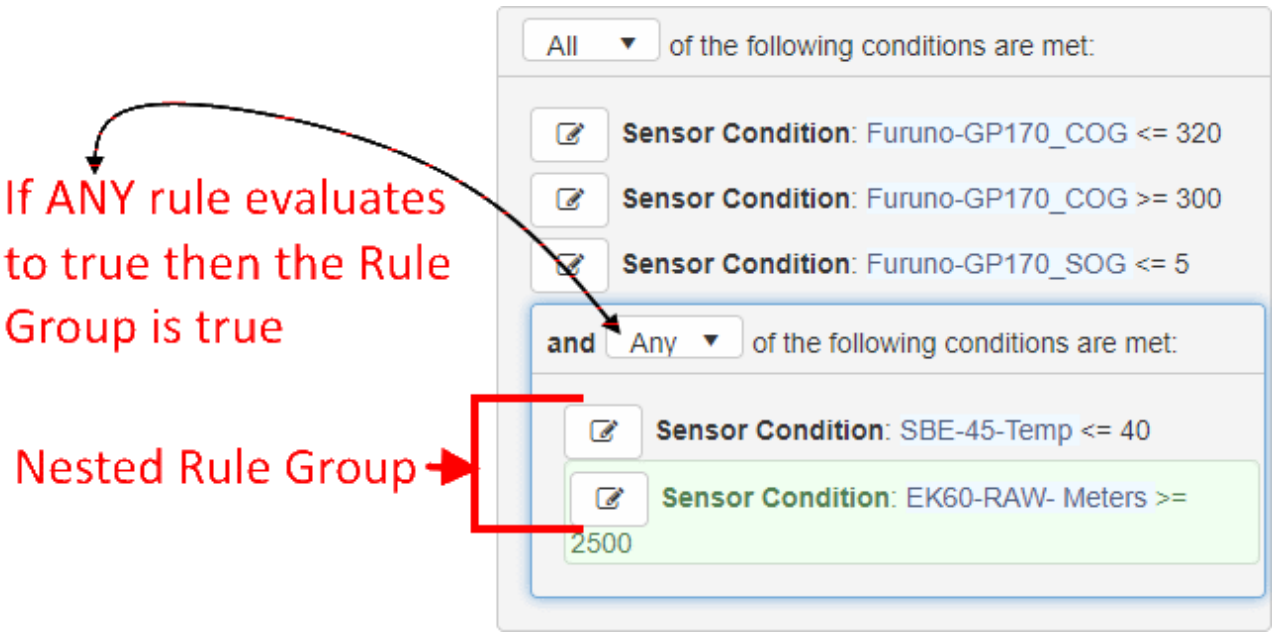
The nested *Rule Group* has its logical operator set to ANY, this means that if EITHER of its two rules evaluates to *true* then the entire group evaluates to *true*.

The concept is clarified below. The root *Rule Group* has it's logical operator set to ALL. That means ALL rules and first level nested *Rule Groups* must be *true* for the trigger to fire. If the COG is greater than 320 then the result is *false*, even if everything else is *true*. If the nested *Rule Group* returns *false* (both it's rules evaluate to *false*) then the root *Rule Group* also returns *false*. ALL it's immediate children MUST return *true* for the root to be *true* as it's logical operator is set to ALL.



In this example there is a single nested *Rule Group*. In reality you can have as many nested *Rule Groups* as you need, and the nested *Rule Groups* can have nested *Rule Groups* of their own down to any level you desire.

⚠ In practicality nesting to many levels will most likely lead to confusion and is not recommended.



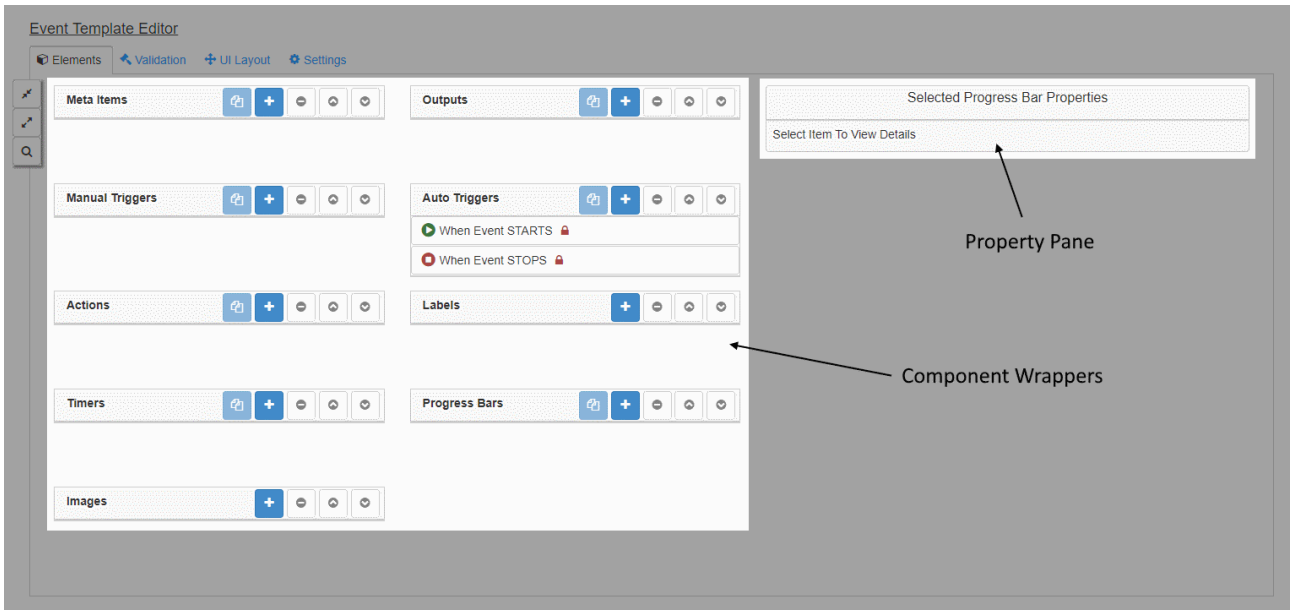
The nested *Rule Group* above will evaluate to *true* if the Temp value is less than or equal to 40, it will also evaluate to *true* if the EK60 depth is greater than or equal to 2500 meters. If ANY of it's rules evaluate to true then the entire [nested] *Rule Group* evaluates to *true*. If you want BOTH conditions to be *true* before the *Rule Group* evaluates to *true* then you must change it's logical operator to ALL.

Event Template Editor - Elements Tab

Elements

Most likely the majority of your time will be spent in the Elements tab. Unfortunately the Elements tab is also where the majority of the complexity is focused. This is where you define all the components of your event and establish the relationships between them.

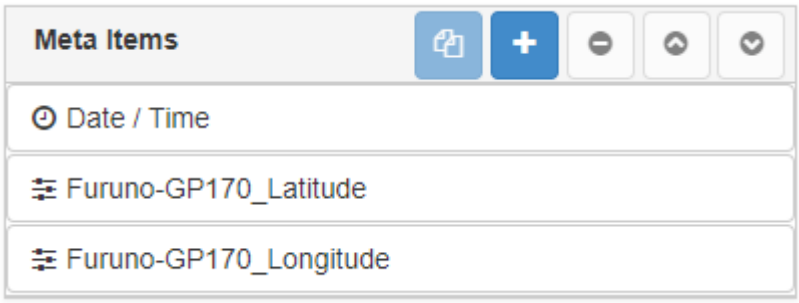
The Elements tab consists of multiple component wrappers on the left with a *Property Pane* on the right.



The general workflow is to click the *Create* button above any component which will create a new instance of that component. It will be automatically selected and it's properties will load into the *Property Pane* for you to edit.

Component Wrappers

In an Event template there are four major elemental components: Meta-Items, Outputs, Manual Triggers and Actions. There are a few others which will be explained later but these are the four you will be primarily dealing with. Each element component has a similar look and feel consisting of a toolbar and a list of instances below it.



For example here is the Meta Items component wrapper with 3 Meta Item components under it.

Clicking on a component instance in the list loads it for editing. Any properties associated with the selected component will show up in the *Selected Component Properties* pane for you to change as you see fit.

> The top toolbar can be clicked to expand or collapse the component.

The toolbar has some very straightforward options, some of which may be disabled until a certain state is achieved (e.g. you cannot *Copy* a component until you have selected the component you wish to copy). Hovering over an enabled button will provide a tooltip explaining it's purpose.

 Copy the selected component

 Create a new instance of the component and add to the bottom of the list

 Remove / Delete the selected component

 Move the selected component UP in the list (may effect display order if *Auto Layout* is selected)

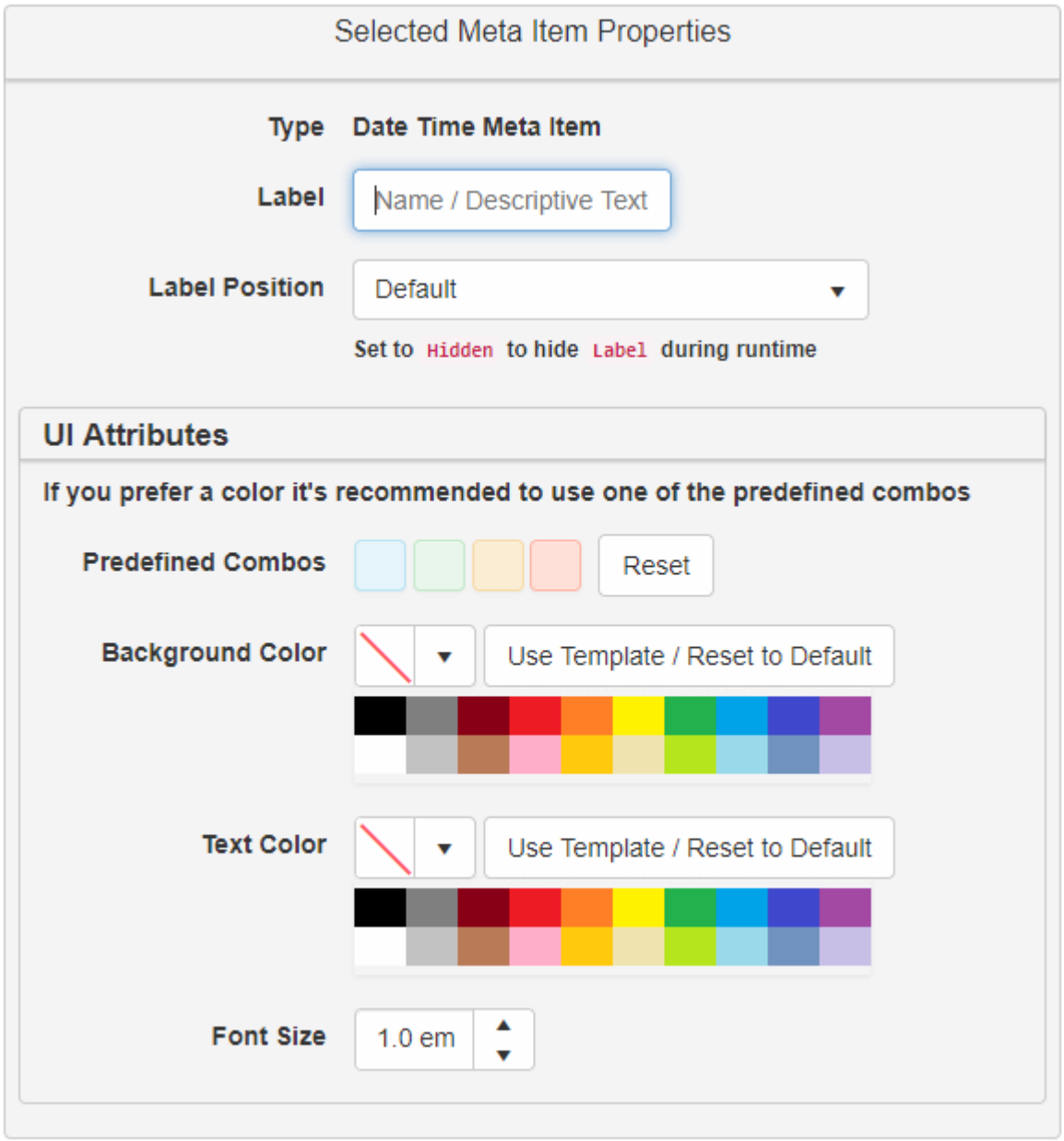
 Move the selected component DOWN in the list (may effect display order if *Auto Layout* is selected)

> You can also move components up and down their respective list using drag & drop.

> On the top left you'll notice a small vertical toolbar. This lets you expand/contract all the element component wrappers at once. It also lets you search all the components for keywords.

Property Pane

The Property Pane is a window which appears on the right of the wrappers and is your main source of interaction with any given component. This is where you set all the details for the selected component and define how it interacts with other components in your template. Each component type may have different properties for you to edit but many of them have a standard set of features as shown below.



When you select a component the property pane will load with all properties relevant to that components type.

The first line displays the type of component you have selected. In this example - Date Time Meta Item

The *Label* input allows you to name this component.

The *Label Position* input allows you to define where the name will be displayed (if at all) on the GUI when this event is run.

A set of UI Attributes is also present for Event components which will be on the display when the event is run. For instance a Meta Item or Label would be displayed, an Output would not.

The main focus here is setting color schemes and changing the font size of the component when running.

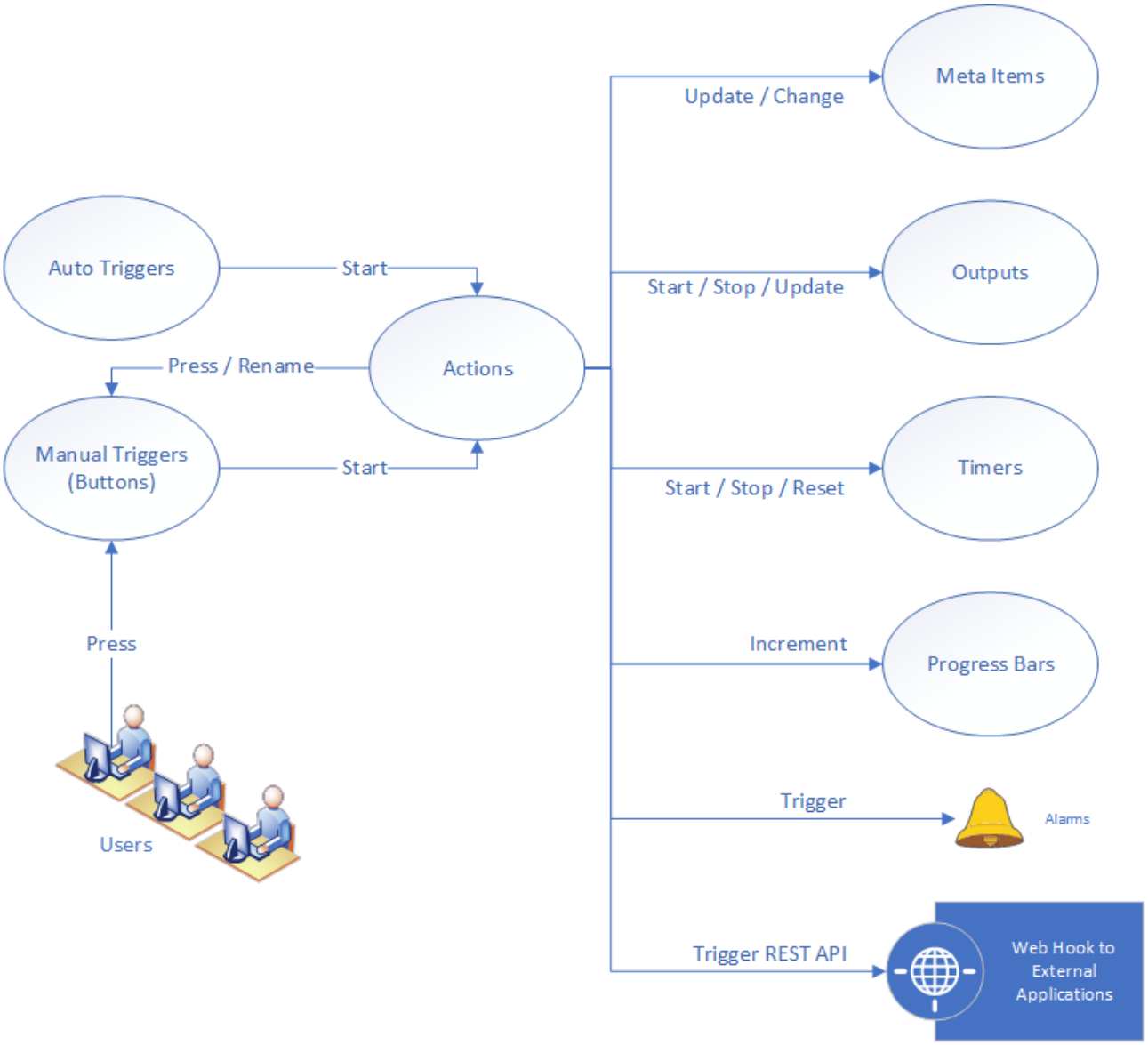
A set of predefined color combinations are provided to you to choose from. Click on the *Reset* button to revert back to no scheme being applied.

Should you choose to modify colors it is HIGHLY recommended that you use one of the Predefined Combos rather than explicitly setting a color. However, if there is a particular color that you feel very strongly about you can always use the color picker or the the palette to set it.

> ⚠ The *Predefined Combo*'s adjust with the selected master theme of the UI. They will change to match everything else when you change it. However, if you explicitly set a color it will lock to that and ignore any master theme changes. This may result in a UI that is hard to read and no longer 508 compliant.

Major Elemental Components - Brief Overview

The following sections give introductory component descriptions and provide explanations of terminology crucial to your understanding. Full details about components and their interactions are provided later in this guide. The different components are closely interdependent, so understanding one component requires the understanding of the others. Therefore, it is suggested that you read through the component types once for an overview, and a second time to fill in details. The following figure illustrates the text.



Meta Items

Meta Items are atomic pieces of data. The heart of each meta-item is its value, which is written to data files or monitored by the Logger during an event.

There are multiple types of meta-items:

- A Date Time Meta Item represents a Date and Time. When updated it gets it's value from the SCS Server's local system clock. As with timestamps throughout SCS it's value is in UTC.
- A Sensor Meta Item gets it's value from a sensor being acquired by the SCS Acquisition Service. When updated it will get the latest value and store/display it.
- A Manual Meta Item must be entered by you from the keyboard during or before an Event. The template editor allows you to define different data types (numbers and strings), place limits on the range of values, or define a drop-down list of values from which the user must choose during run-time.
- A Global Meta Item get's it's value from a set of meta data provided to ALL events at any given moment. While these can be included in any event they can only be changed by users with appropriate rights as they will then impact all events referencing them.

- A Summary Meta Item is computed automatically by the Event Logger Service at the end of an event. Using the Builder, you define the type of value to be calculated (average, minimum, time duration, etc.), and which Meta Items or sensor data form the basis of that calculation.

Outputs

The function of an Output is to log data (Meta Items or sensor values). You can set up multiple Outputs, writing different sets of data to different files during an Event. Individual Meta Items / sensor values are known as the elements of the Output. During an event the Logger logs its elements as a set of time stamped values.

Outputs may be individually started, stopped and restarted during an event. Once started, logging may be continuous (elements are written periodically) or a snapshot (only one set of elements is written, then the Output is stopped).

Manual Triggers

Manual Triggers appear as buttons on the Logger's GUI. Typically you will click on a button in response to external events that you have predefined, such as "Net in water."

When you define a Manual Trigger in the editor you must assign at least one Action sequence to it. Then, when you click on it during a run of an Event the Logger kicks off each Action sequence you have linked to it.

> You may also arrange for a Manual Trigger to be automatically clicked for you by an Action which in turn can be kicked off by any Manual Trigger or an Auto Trigger.

Actions

Action sequences are a powerful new feature of Events. Essentially it's a predefined list of "things to do" such as update a Meta Item, start an Output or fire off an Alarm. Action sequences are started via a trigger, either manually via a button (Manual Trigger) or automatically when a condition is met (Auto Trigger).

Elemental Components - Detailed

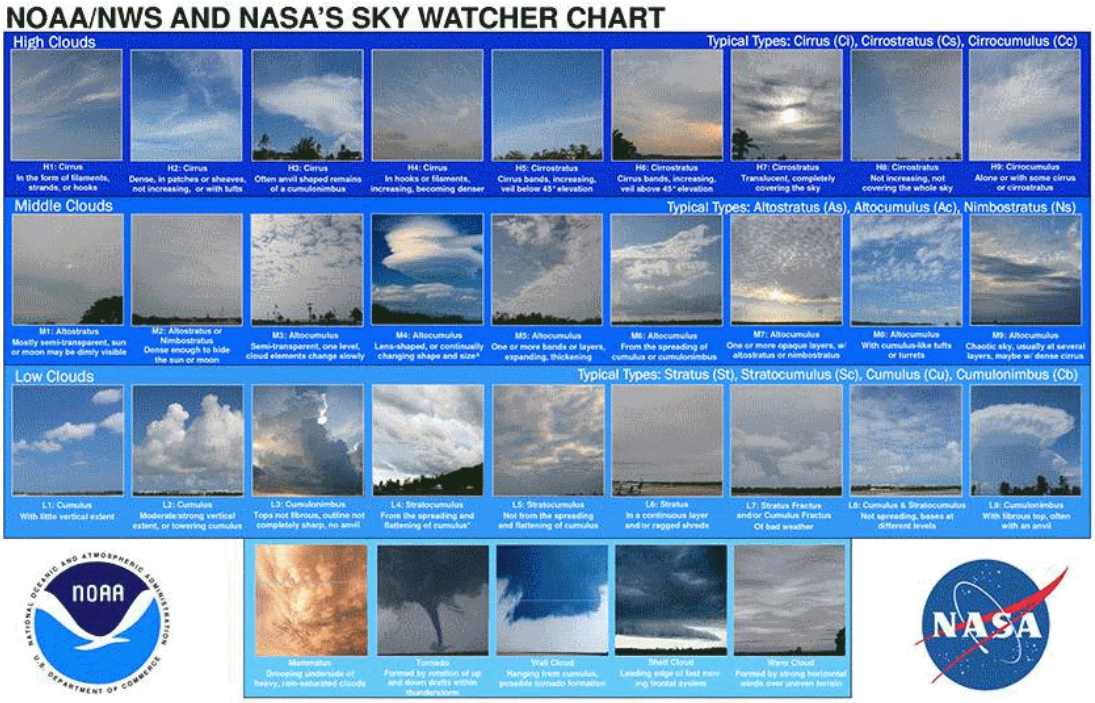
- Action Sequences
- Alarms
- Automatic Triggers
- Images
- Labels
- Manual Triggers

- Meta Items
- Outputs
- Progress Bars
- Timers

Images

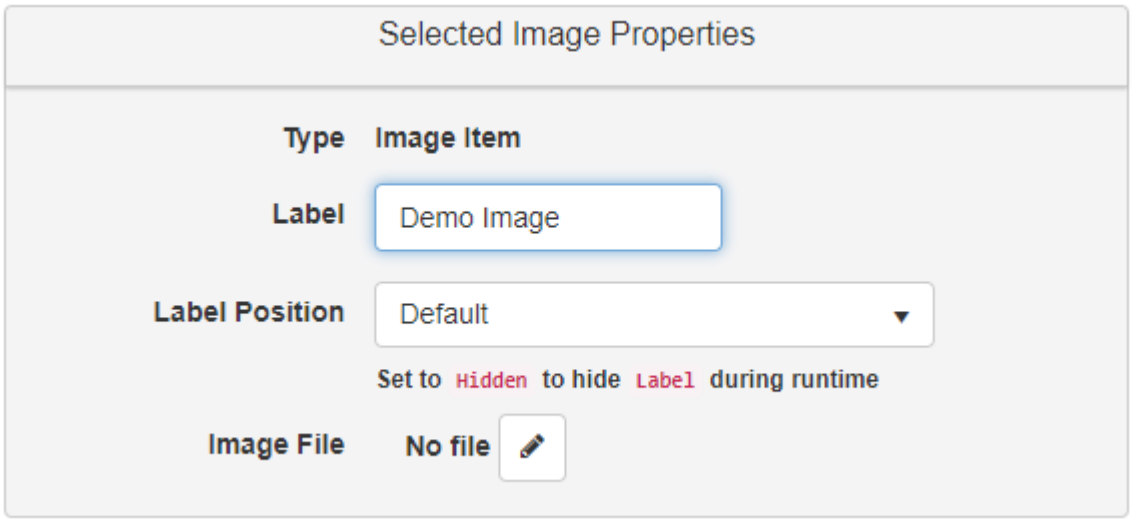
Images do not actually interact with or have any impact from a technical perspective on the Event. Their main purposes is to provide users who are running the event with visual clues or cheat sheets to assist them directly on the event screen.

For instance, your average person might be asked to manage an event which records the various cloud coverage throughout a cruise. Perhaps they have been doing this for decades and know their stuff, but maybe not. Providing an image such as the one below right in the event itself might assist them in their selection and make the output of the event more accurate.



Sea state charts, relevant bird/mammal species images and so forth all are reasonable justifications for putting an image into an event. While images are in no way required in some use cases they may provide help that cannot be provided via any other means.

The

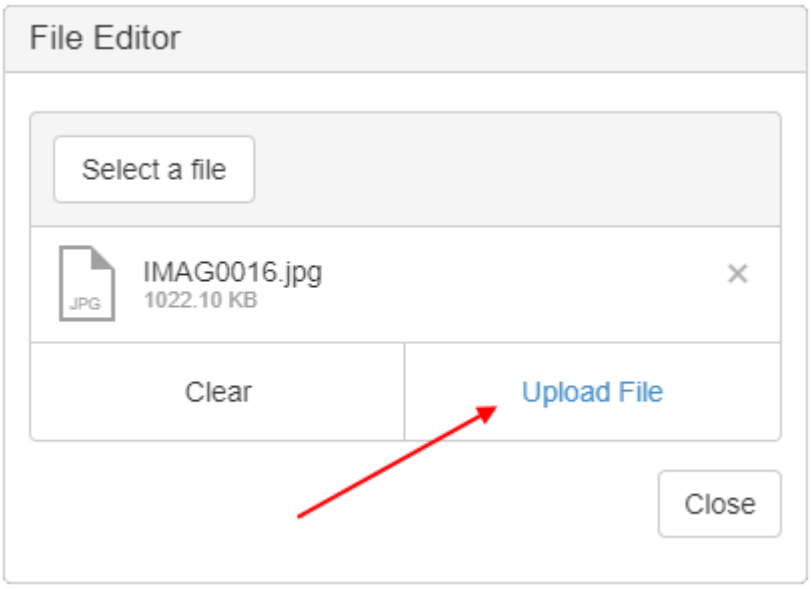


Images *Property Pane* only really has one unique option, that being the image file itself.

The *Label* will appear as part of the tooltip on the UI. *Label Position* is ignored in this case.

To assign an actual image to the component you must click the *Edit* button next to the *Image File* property. This will bring up the standard SCS file upload popup box.

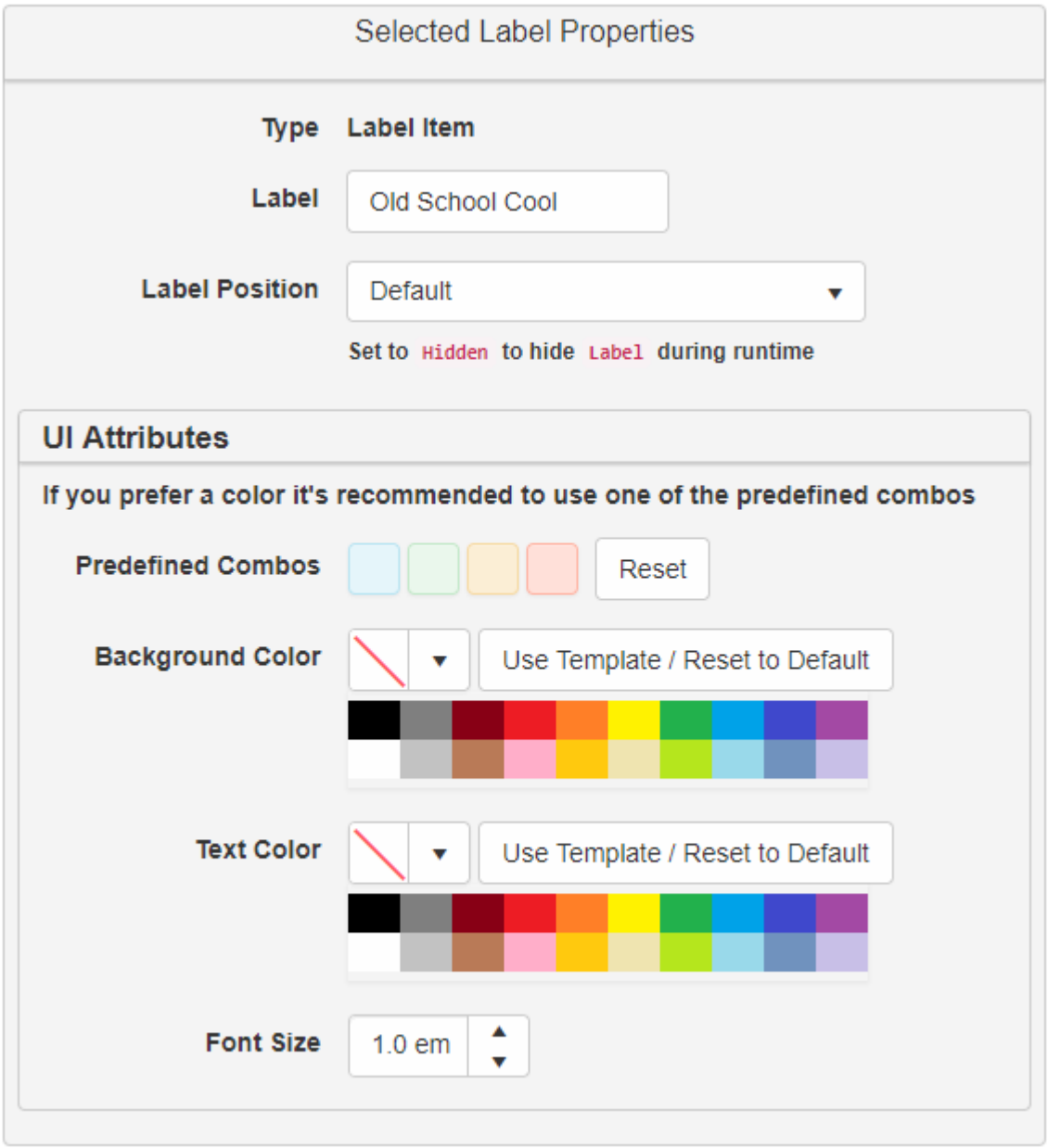
Select your image file and be sure to click the *Upload File* button before closing the dialog!



Labels

Labels are the most simple component in an Event. Similar to Images, they do not actually interact with or have any impact from a technical perspective on the Event. All Labels do is provide a means of adding

some hard-coded text to the Event UI. This can be useful for clarifying your UI when a *manual layout* is used.



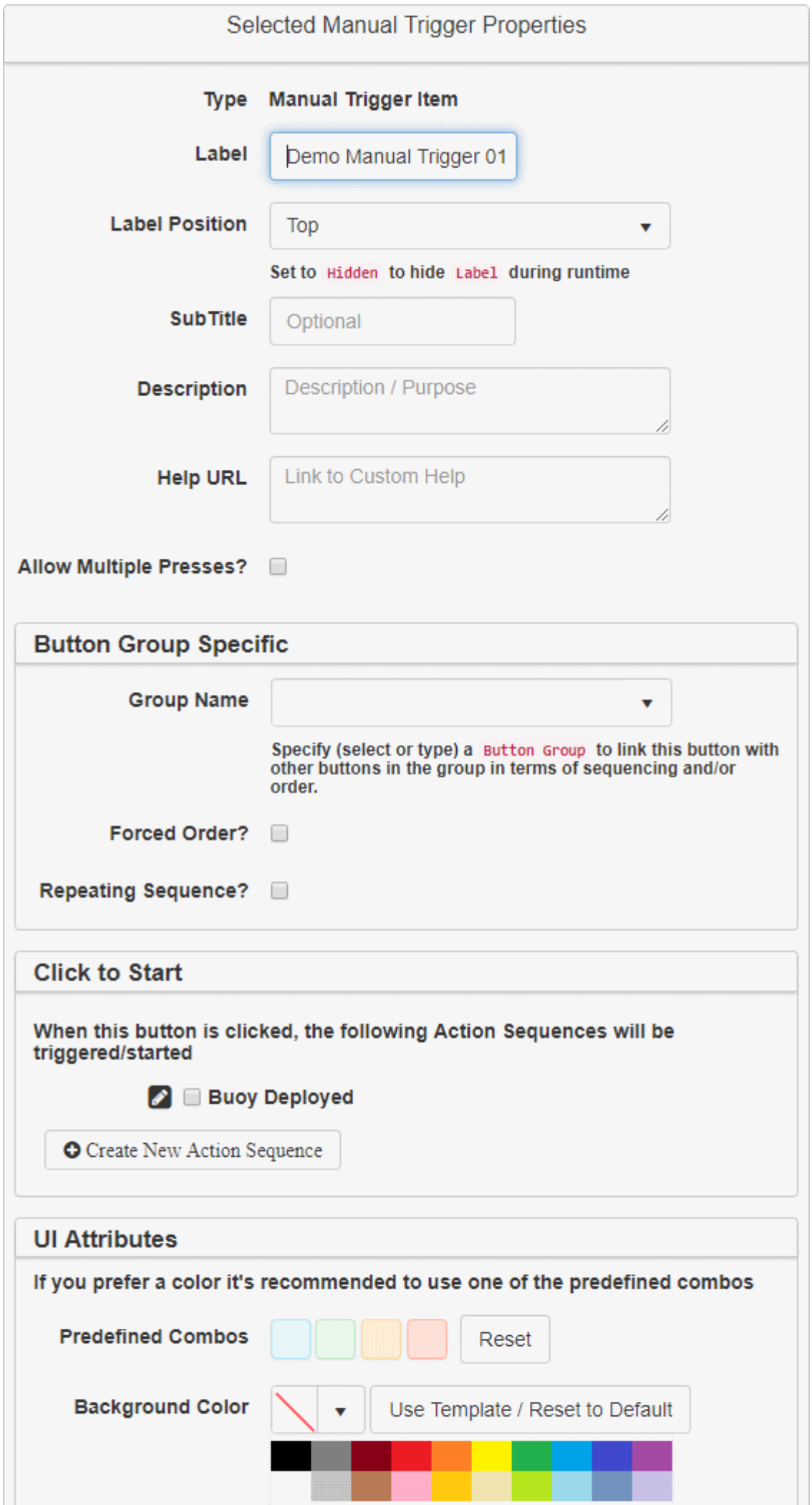
Labels are extremely basic and have no unique properties.

The value you place in the *Label* property is simply displayed on the UI.

Manual Triggers

Manual Triggers provide a means for a user running an Event to manually trigger an Action Sequence. Manual Triggers are displayed on the Event UI as buttons, the *Label* you provide shows up inside the button so the user has an idea what that buttons purpose is. For instance you might have a Manual Triggers labeled "Buoy Deployed", a user running the event (with perhaps a quick briefing) will most likely understand that when a buoy is deployed that button should be pressed.

Manual Triggers in the modern Event framework have many new features, all of which are optional. The only real requirement you need for a valid Manual Triggers is a *Label* and one or more Action Sequences you want the trigger to start.



The Property Pane for

a Manual Triggers has many more options then most other components.

The *Label* is displayed inside the button, it's relative location is refined via the *Label Position* property.

An optional *SubTitle* property allows you to further clarify the *Label*. If provided it will be also be displayed inside the button, though at a smaller font than the *Label*.

An optional *Description* will set the tooltip of the button. If a user hovers their mouse over the button this is the test that will appear for them.

Optionally, should you have an external help system you may provide a link to any help documentation you want to provide for this button via the *Help URL* property.

The *Allow Multiple Presses* property determines whether the button is allowed to be pressed multiple times or not. If left unchecked then once a button is pressed it is disabled and may not be pressed again. If this is checked then a user can click the button over and over again.

Button Groups allow sets of related Manual Triggers to be grouped logically together. Details on this concept and functionality can be found below. To create a new group simply start typing a group name into the dropdown box, to join an existing group choose it from the dropdown.

Similar to Auto Triggers, Manual Triggers are simply a means to fire off one or more Action Sequences. Check all sequences you want the button to trigger when pressed.

Button Groups

The idea behind *button groups* is to take a set of buttons which have some connection or interdependence and put them in a logical container. For instance lets say you have 4 objects you want to deploy off the back of the ship and create a Manual Trigger representing each of them:

If these 4 objects can be deployed in whatever order you want then no group is needed. However, if they have to be deployed in order then this is where a *Button Group* becomes useful.

If you put all 4 of these triggers into the same *Button Group* and check the *Forced Order* checkbox for each of them then buttons 2,3 and 4 will be disabled upon startup. When button 1 is clicked, it will become disabled and button 2 will become enabled. When button 2 is clicked it will become disabled and button 3 will become enabled, etc. The currently enabled button continues to move on down the line until object 4 is deployed. At this point all 4 buttons will be disabled and no more objects can be deployed.

If you would like to have the sequence repeatable then you can click the *Repeating Sequence* checkbox for each of the buttons (and ensure *Allow Multiple Presses* is checked!). When the last button in the sequence is clicked the sequence starts back over (e.g. button 1 becomes enabled again).

To summarize the above, for all buttons in a group with the *Forced Order* flag set:

Forced Order: Entered into a press order, no button can be pressed out of order

Repeating Sequence: When the last button in the press order is pressed the order is reset and the first button (with the MultiPress flag set) is enabled, starting the press sequence over again.

If a Manual Trigger is part of a group but does not have the *Forced Order* flag set it can be clicked regardless of which button current is active in the press order.

If a Manual Trigger is part of a group but does not have the *Allow Multiple Presses* flag set then it cannot be clicked again, even if part of a *Repeating Sequence*

If a Manual Trigger is part of a group but does not have the *Repeating Sequence* flag set then it cannot be clicked again, even if *Allow Multiple Presses* is checked.

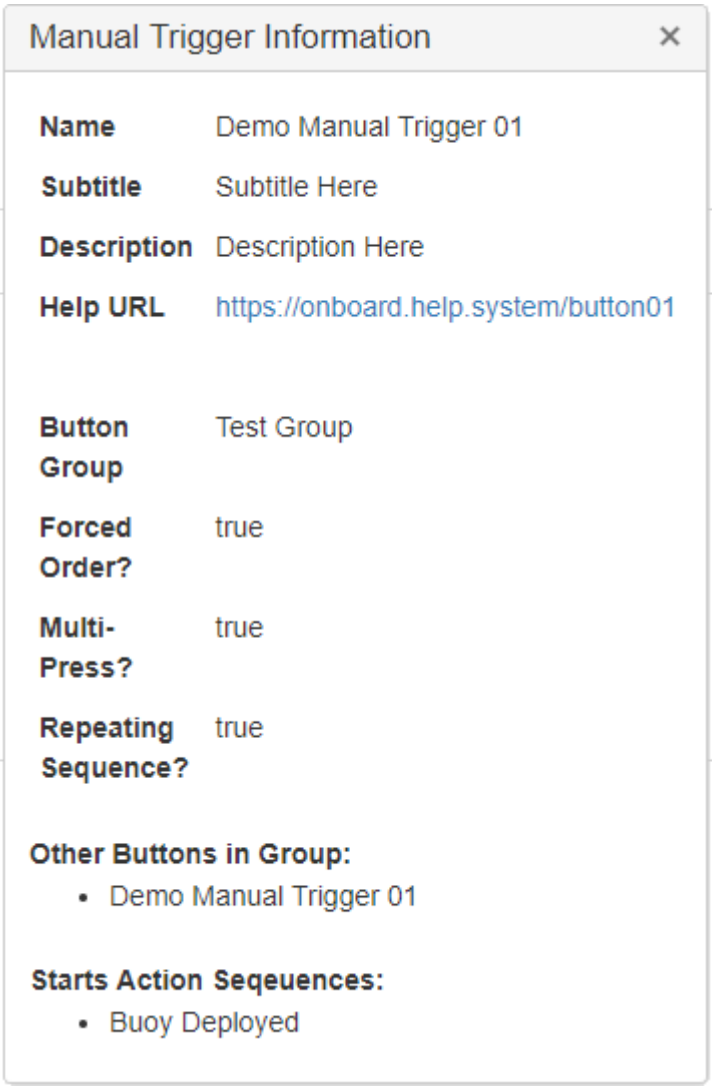
⚠ The combinations of *Button Groups* and which flags are set are critical to Manual Triggers operating correctly. It is highly advised you test any event you create prior to conducting a mission operation to ensure it works as expected!

Manual Trigger UI

A Manual Trigger shows up in the Event UI as a button. On the left side of the button you will notice icons (if any) which indicate which flags and attributes have been set for that button.

The *Label*, an optional ***SubTitle*** and optional *ProgressBar* can also be seen inside the button.

Right clicking on the button brings up a context menu which will provide additional details, a link to any optional help and a history of all the times the button was pressed during the run of that event.



Meta Items

Meta Items are atomic pieces of data. The heart of each meta-item is its value, which is written to data files or monitored by the Logger during an event.

> All Meta Items are updated via an action inside an Action Sequence.

Date Time

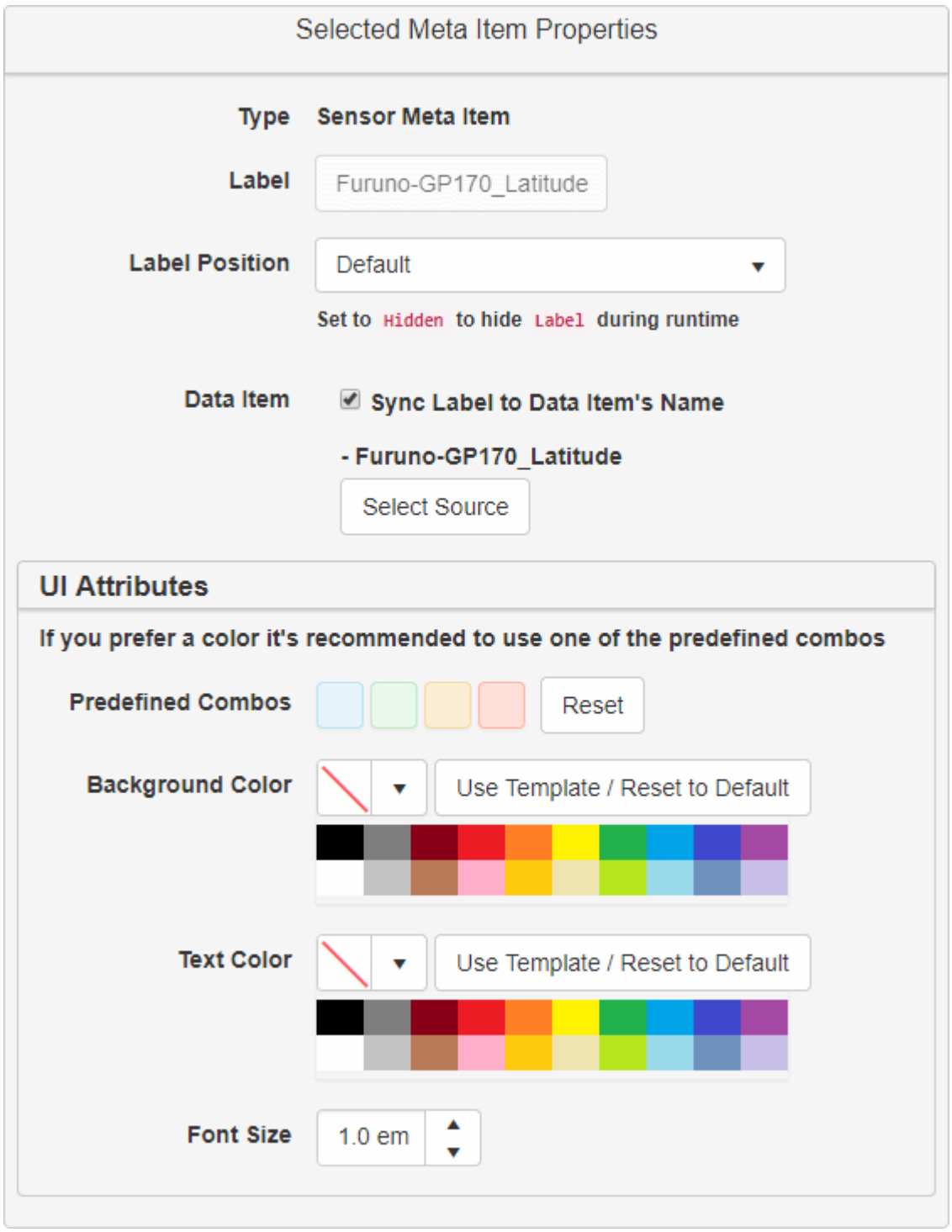
A Date Time Meta Item represents a Date and Time. When updated it gets it's value from the SCS Server's local system clock. As with timestamps throughout SCS it's value is in UTC.

The only option to set for a Date Time Meta Item is the *Label*. Most things in Events are time stamped independent of any Meta Item, however if you want to note the time on the UI when something last occurred this can provide a means of doing so.

See Also: Action Sequences - Meta Item Update

Sensor

A Sensor Meta Item gets it's value from a sensor being acquired by the SCS Acquisition Service. When updated it will get the latest value and store/display it.



The *Label* for a Sensor Meta Item can be something you type yourself or you can link it to the sensor feed supplying the data by checking the *Sync Label to Data Item's Name* checkbox.

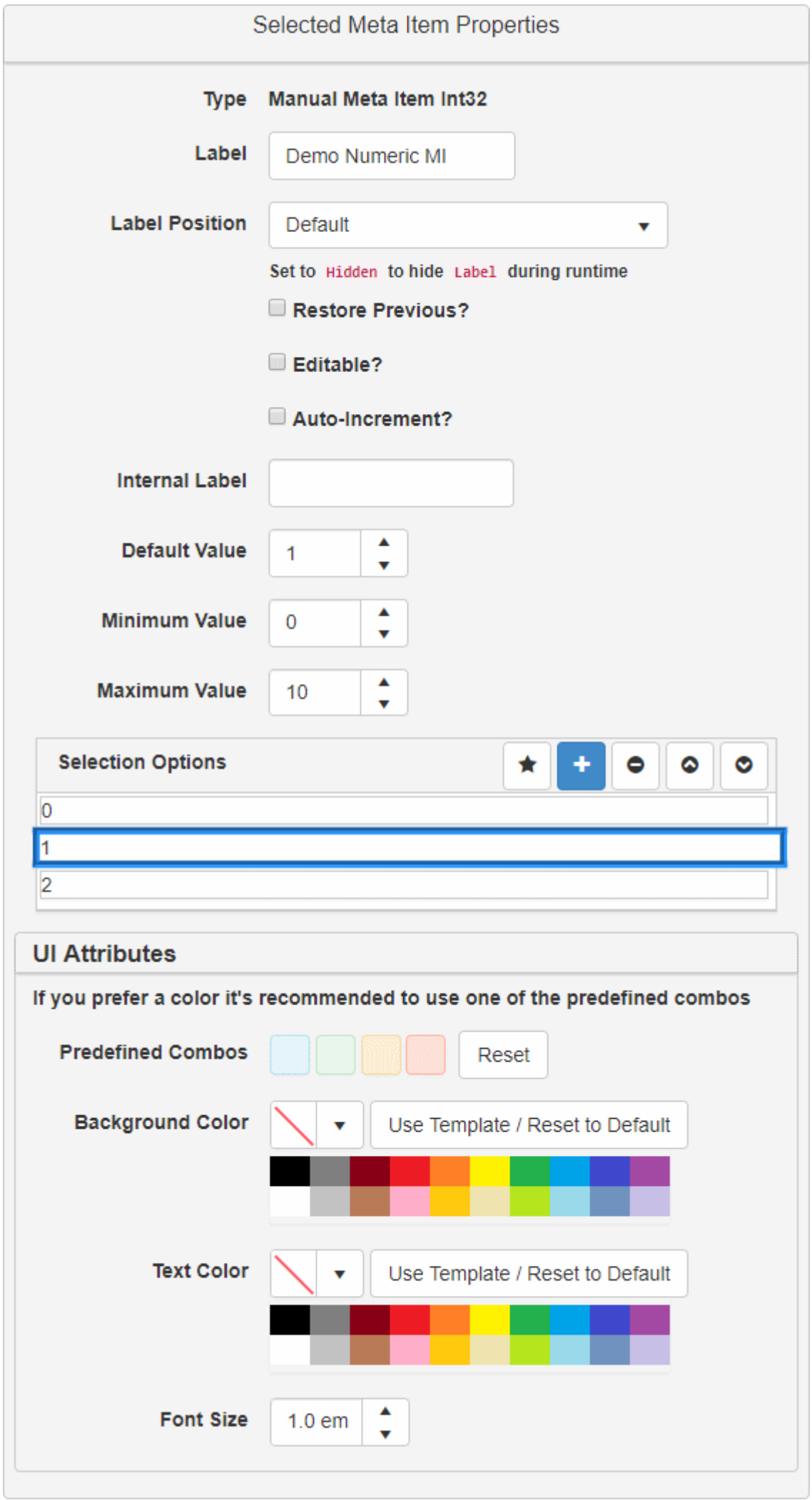
Click the Select Source button to set the sensor feed you wish to associate with this component

See Also: Action Sequences - Meta Item Update

Manual

A Manual Meta Item must be entered by you from the keyboard during or before an Event. The template editor allows you to define different data types (numbers and strings), place limits on the range of values, or define a drop-down list of values from which the user must choose during run-time.

Numeric



The *Restore Previous* checkbox tells this Meta Item to use the last value recorded from the last run of the event. For instance if the last time the Event was run this Meta Item was at '15' then the next time the Event is started the Meta Item will continue and start at '15..

The *Editable* flag determines whether a user can manually change the value of the Meta Item.

The *Auto-Increment* flag tells the Meta

Item to automatically add 1 to it's value for each run of the event.

An optional *Internal Label* can be displayed inside the input box to help the user understand what they are entering (eg 'seconds' or 'meters', etc)

The *Default Value* is what the Meta Item will be set to barring any other logic (see above checkboxes) override it.

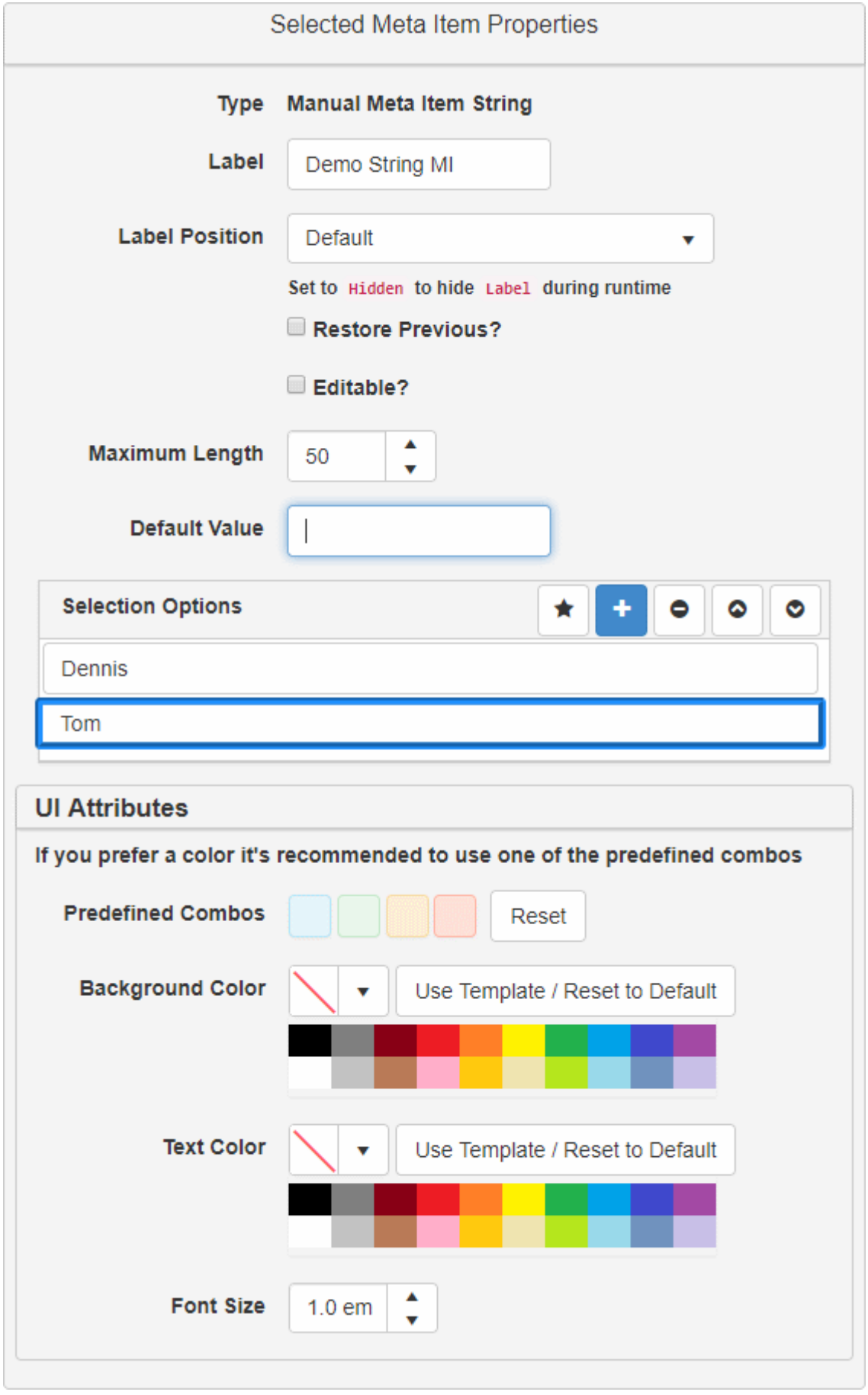
The *Minimum Value* is the lowest number the Meta Item can be set to.

The *Maximum Value* is the highest number the Meta Item can be set to.

If you would prefer the user to choose from a list of numeric options rather than enter their own values you can provide those options here.

See Also: Action Sequences - Meta Item Increment

String

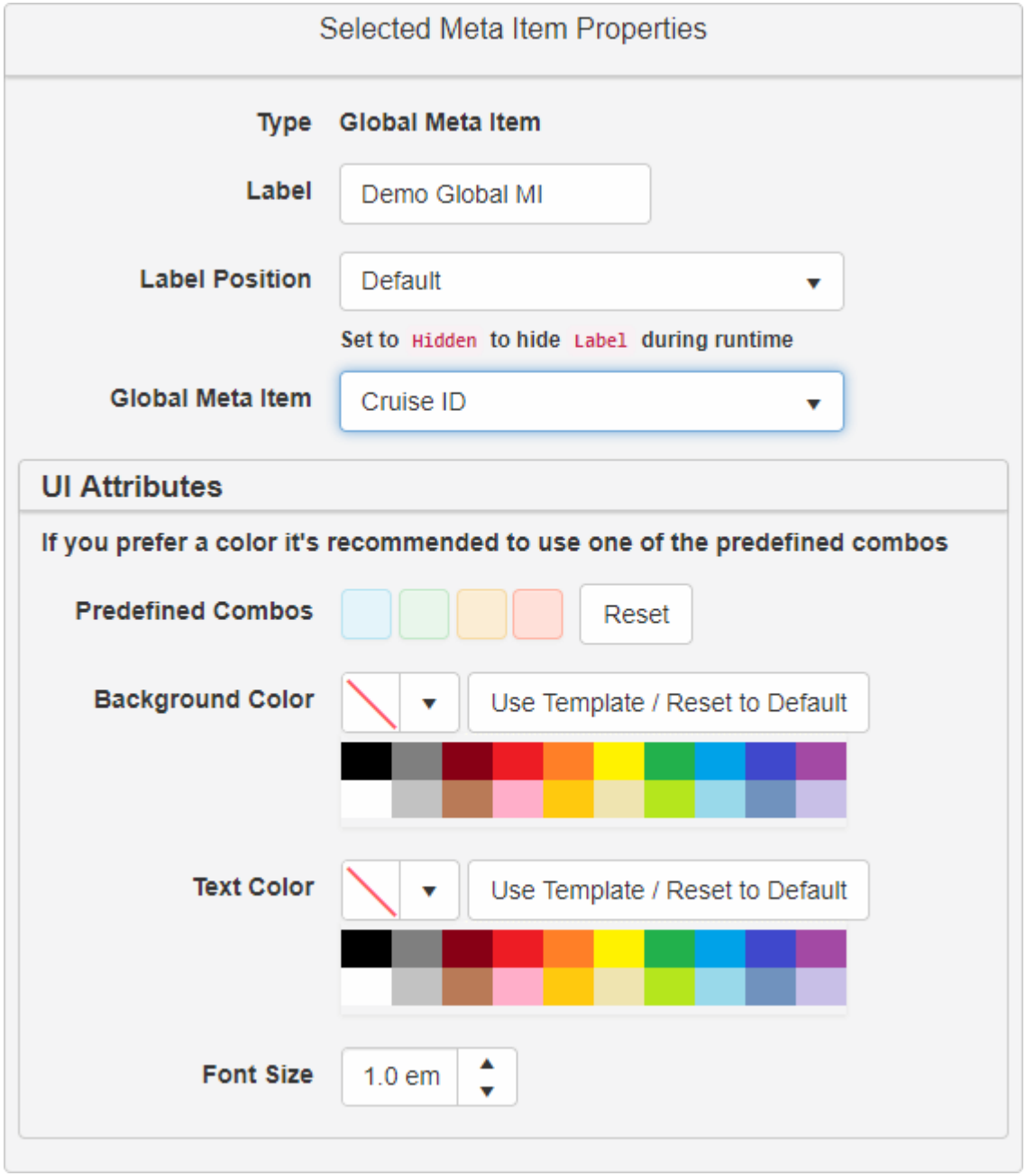


Options for the string variant are very similar to the numeric one, see above for details.

A *Maximum Length* can be set preventing the user from entering a string value with more than 'x' characters.

Global

A Global Meta Item get's it's value from a set of meta data provided to ALL events at any given moment. While these can be included in any event they can only be changed by users with appropriate rights as they will then impact all events referencing them.



The only real option for a Global Meta Item is to choose which one you want this component to point to. This dropdown is populated from the master list which is managed via the primary *Event Management* page.

See Also: Global Meta Item Management

See Also: Action Sequences - Meta Item Update

Summary

A Summary Meta Item is computed automatically by the Event Logger Service at the end of an event. Using the Builder, you define the type of value to be calculated (average, minimum, time duration, etc.), and which sources form the basis of that calculation.

Summary Meta Item - Time Duration

Result is a quantity of time / duration. You can choose between using Meta Items or Outputs as your source.

Meta Item Source

Result is the difference between the LAST recorded value for the Start Meta Item and the LAST recorded value for the End Meta Item

> If the same MI is selected for both start/stop then the duration will be the difference between it's first and it's last value when the event concludes.

Output Source

Result is the amount of time between the first and the last 'writes' to the selected Output.

Summary Meta Item - Average Value

Result is an average value of a numeric source. You can choose between using Meta Items , Outputs or a direct sensor feed as your source.

> If you choose a direct sensor feed as your source it will average every value from the data field logged by ACQ over the course of the entire event regardless of how or even IF it is being used elsewhere in the event.

Summary Meta Item - Polar Average

Same as normal average above except using a polar coordinate system.

> Use this when trying to average 'circle' type values such as COG or Heading.

Summary Meta Item - Minimum Value

Result is the minimum value of a numeric source. You can choose between using Meta Items, Outputs or a direct sensor feed as your source.

> If you choose a direct sensor feed as your source it will take the minimum value from the set of every value of the data field logged by ACQ over the course of the entire event regardless of how or even IF it is being used elsewhere in the event.

Summary Meta Item - Maximum Value

Result is the maximum value of a numeric source. You can choose between using Meta Items, Outputs or a direct sensor feed as your source.

> If you choose a direct sensor feed as your source it will take the maximum value from the set of every value of the data field logged by ACQ over the course of the entire event regardless of how or even IF it is being used elsewhere in the event.

Summary Meta Item - Rhumbline

Result is the distance between two points Δs, measured along a loxodrome, is simply the absolute value of the secant of the bearing (azimuth) times the north–south distance (except for circles of latitude for which the distance becomes infinite). A rhumbline appears as a straight line on a Mercator projection map.

-Wikipedia!

In other words, this summary calculation will return the distance between a starting point and an ending point as if the ship went 'straight' from one to the other. You must use a starting Lat/Lon Meta Item and an ending Lat/Lon Meta Items as your source.

Summary Meta Item - Trackline Length

Result is the distance traveled over a course of time. You can choose between using Outputs or a direct sensor feed as your source.

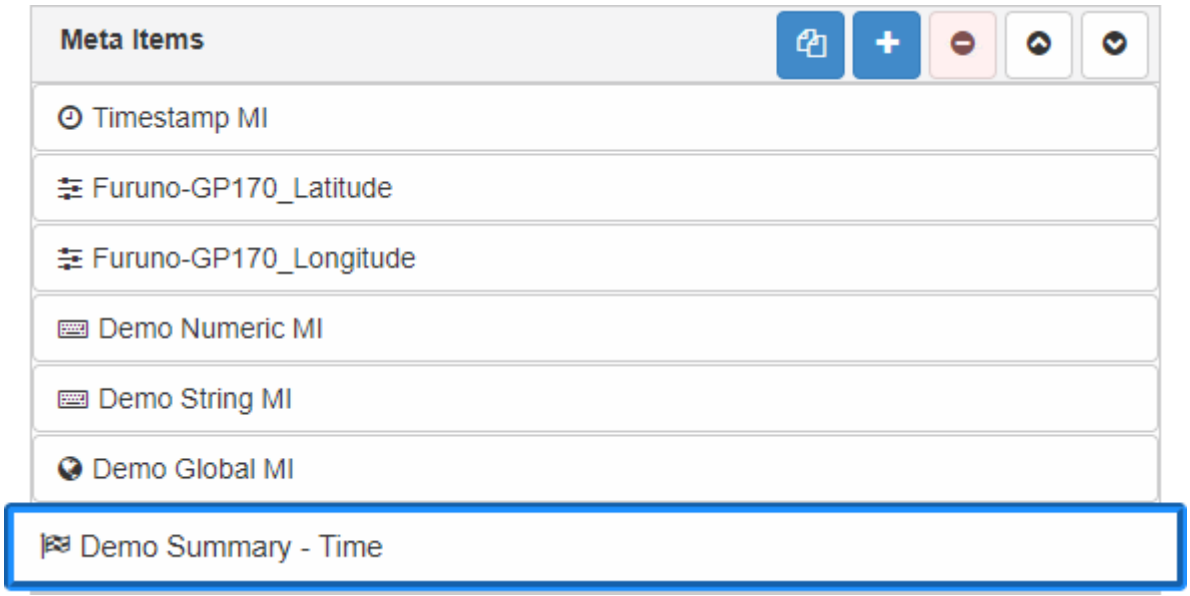
> If you choose a direct sensor feed as your source it will take the minimum value from the set of every value of the data field logged by ACQ over the course of the entire event regardless of how or even IF it is being used elsewhere in the event.

> If you choose to use an Output then the length will be over the course of the Output itself. If the Output doesn't start when the event starts, stops before the event stops, etc, then it will NOT represent the length traveled over the course of the event. If you want a trackline length for the whole event either use a sensor feed or ensure your Output starts and ends at the same time as the event (consider using the two prebuilt Auto Triggers to be safe).

> The difference between a Rhumbline and a Trackline is that the trackline takes into account the curves, transects and the zip-zagging of the ship whereas the Rhumbline ignores all the meandering between the start/end coordinates and assumes a straight path.

Icons

When looking at the Meta Items you quickly determine their type by looking at the icon set on the left hand side of each component. Hover your mouse over the icon for feedback on what it represents.



Outputs

The function of an Output is to log data (Meta Items or sensor values). You can set up multiple Outputs, writing different sets of data to different files during an Event Individual Meta Items / sensor values are known as the elements of the Output. During an event the Logger logs its elements as a set of time stamped values.

Outputs may be individually started, stopped and restarted during an event. Once started, logging may be continuous (elements are written periodically) or a snapshot (only one set of elements is written, then the Output is stopped). Both types of Outputs are controlled (Start/Stop/Update) via the *Output* Action inside an Action Sequence.

See Also: Action Sequence - Output

The Property Pane for an Output contains a few unique inputs.

First you must choose whether you want this Output to continuously write data at a set interval. If so choose the *Continuous* Output Mode and supply an *Interval* (eg if you wanted it to log its data 1x every second then you should increment the seconds field to 1).

If you would rather the Output only record a line of data when prompted to do so then choose *Discrete / Snapshot*.

As stated above, to start or stop a *Continuous* Output or update a *Discrete / Snapshot* Output you must use the *Output* action inside an Action Sequence.

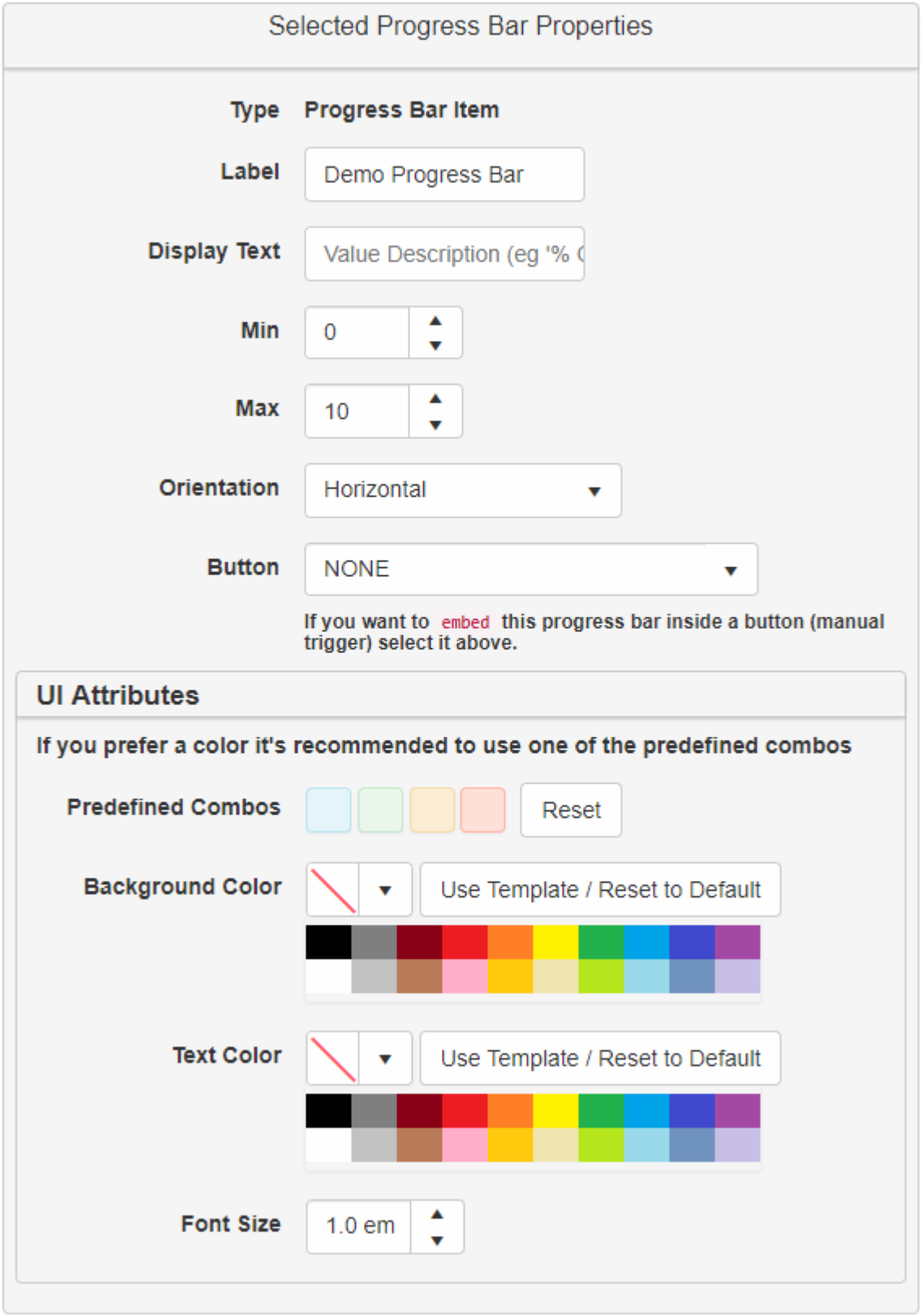
Above you decide on how often you want the Output to log. In the *Data to Log* section you select what exactly it is you want the Output to record.

Click the *Manage* button to bring up lists of respective data items to add to this output. When the Output is started it will log each of these items at the interval you specify until it is stopped. Or if it's a Discrete/Snapshot Output it will log each of these data items every time an Update action targets it.

Progress Bars

Progress Bars can be added to Events to indicate a numerical progression towards a goal. The concept of a progress bar is nothing new and they can provide useful feedback for the users running your event. For instance, lets say you plan on deploying 10 buoys off the back of the ship. Every time a buoy is deployed you can increment a progress bar so users can see how many have gone over and how many are left to do.

Progress Bars can also be embedded in buttons, this further visually relates progress with a specific action. For instance, placing the above mentioned Progress Bar into a Manual Trigger named "Deploy Buoy" makes it easy to find and rather obvious what it's purpose is.



The *Label* you provide will show up as the tooltip for the Progress Bar.

The *Display Text* will show up inside the Progress Bar to provide additional feedback to the user.

A *Min* value can be set, this is where the Progress Bar will start.

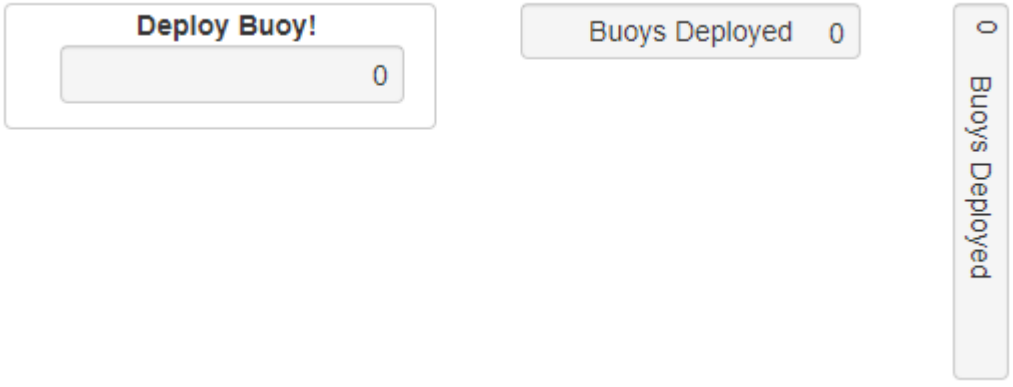
A *Max* value should be set, this is where the Progress Bar will end (100%).

You can orient the Progress Bar horizontally or vertically on the UI.

If you choose to embed the Progress Bar into a Manual Trigger then you must select that component here.

Progress Bars are incremented via the *Progress Bar* action inside Action Sequences.

Below are some examples showing the same Progress Bar embedded in a Manual Trigger and with the two orientations.

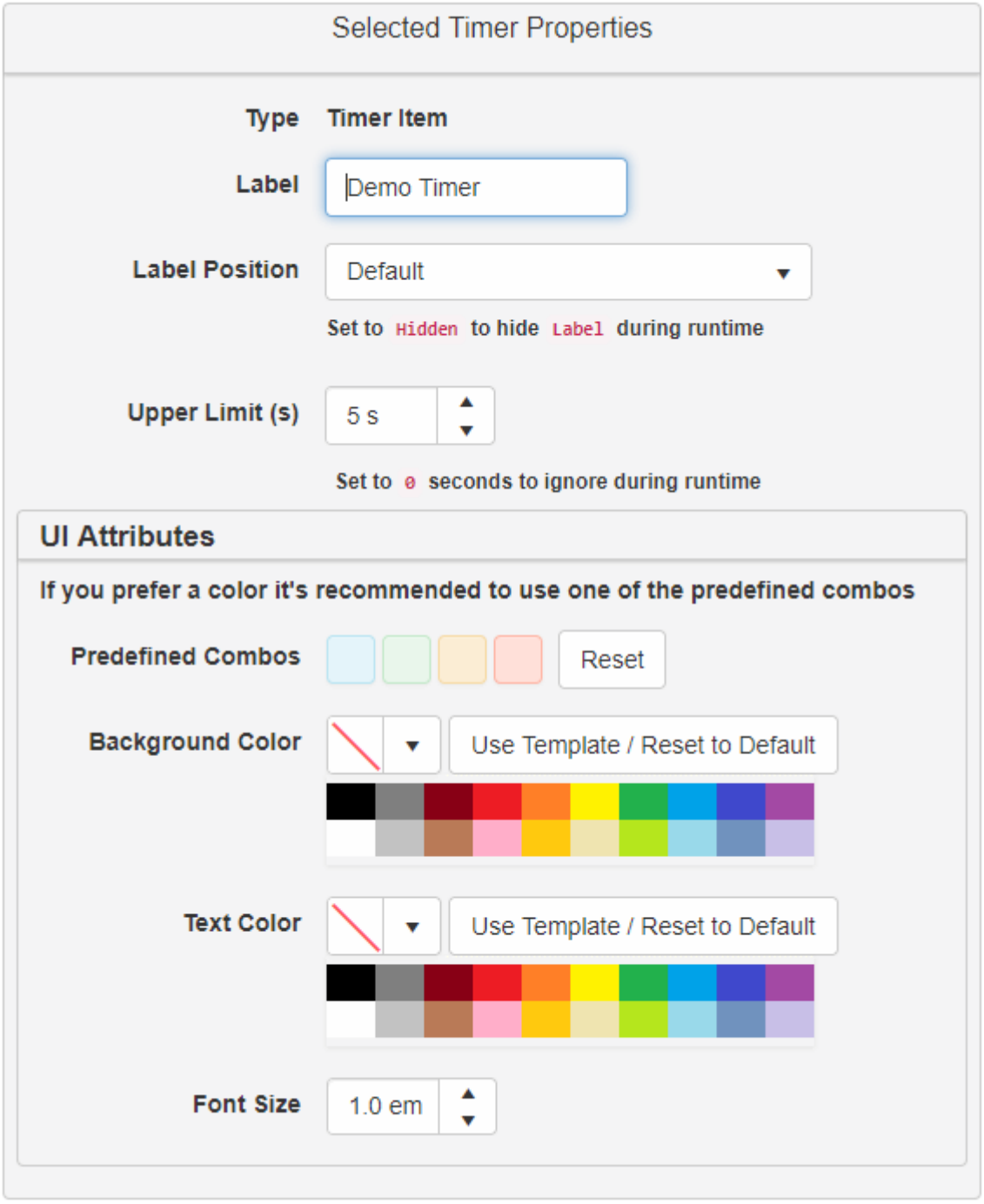


See also: Action Sequences - Progress Bars

Timers

Timers are essentially simply visual stopwatches displayed on the Event UI. Timers can be used to provide visual feedback on any temporal metric a user might be interested in, such as how long the Event has been running, how long it's been since a Manual Trigger was clicked or how long it was between two Manual Trigger clicks, etc.

Timers are controlled via the "Timer" actions inside Action Sequences. They can be started, stopped/ paused and restarted via the respective action types.



The only unique property for

Timers is the *Upper Limit*. The *Upper Limit* is optional, if left at 0 the timer will increment indefinitely as limited by the event's Action Sequences. If an *upper limit* is set then once that limit is reached the Timer will flash and indicate to all users that it has met it's limit.

The *Upper Limit* is measured in seconds.

A Timer is a very basic display (see below). It is simply the *Label* name followed by it's current value (hours : minutes : seconds).

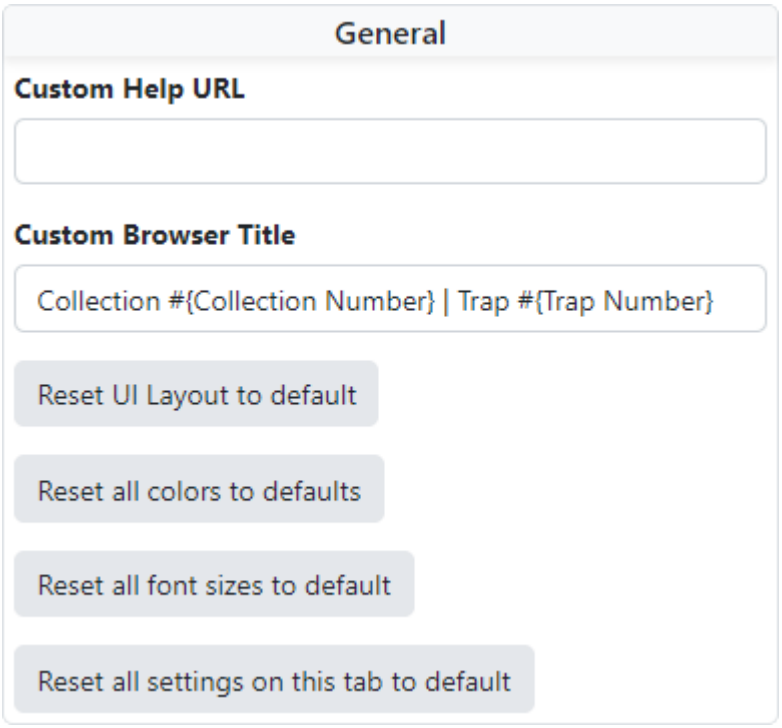
See Also: Action Sequences - Timers

Event Template Editor - Settings Tab

Settings

The settings tab covers general settings for the entire event template. The scope of these items exceeds any one given component. There are a few settings groupings as defined below.

General

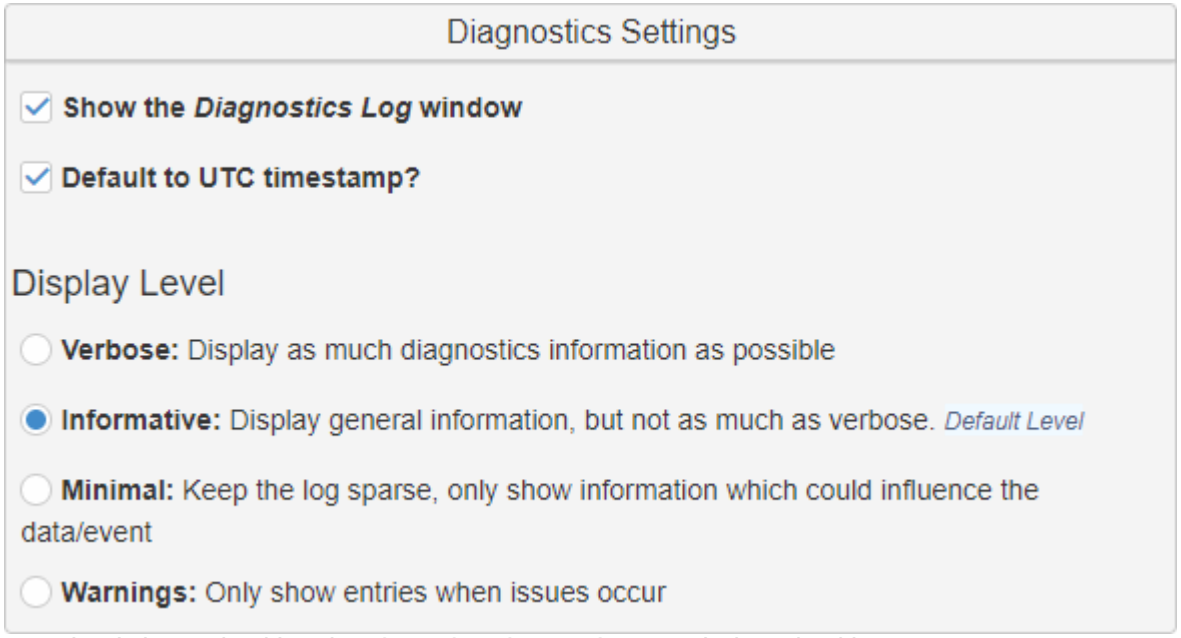


The general settings block allows you to:

- Supply a URL to your custom (if available) help system for overall help with the event
- Customize the tab title which shows up when you run the event
- Reset various settings and UI elements to their defaults.

Diagnostics

During the run of the event lots of system logs are generated to document what is occurring inside the event. Some log entries are more important than others, eg an Output reporting an error is more important than knowing that someone has left the event. These logs can be displayed on the UI so the users participating in the event can see them in real time.



If you don't want to see logs you can hide the window completely by unchecking the *Show the Diagnostics Log* window checkbox.

All times in SCS are recorded and conveyed in UTC, however if you would prefer to view the timestamps in your local time zone you can uncheck the Default to UTC checkbox. This does NOT change how the timestamps are recorded (remains UTC) only how the log window displays them on your browser for THIS event.

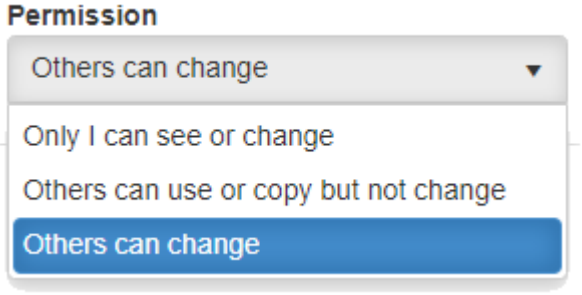
You can also filter the level of information you want to receive. If you only care about warnings and problems set the level to Warnings. If you want to get as much of the log as you possibly can choose Verbose. Note this has no impact on what is actually logged, only what is displayed on the UI.

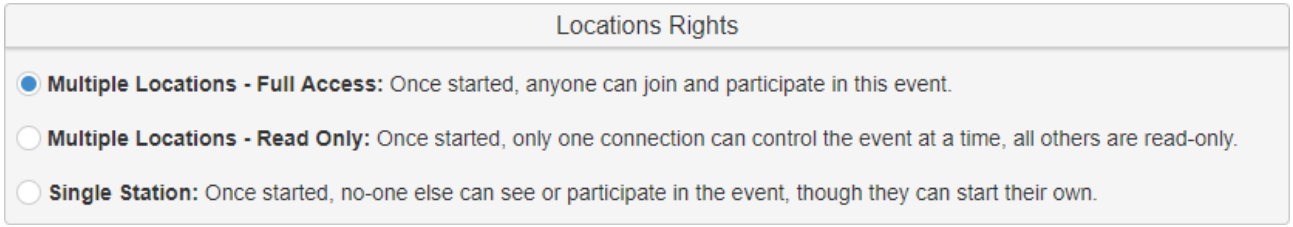
Location Rights

Events in the modern SCS framework are web based. As such Events allow users to participate in the same event from all over the ship, even from shore should you choose to allow this. The idea being that sometimes the bridge, the back deck, those in the lab and so forth all have their part to play in a more complicated Event (like during a trawl). Rather than one person monitoring a radio waiting around and clicking buttons for everyone else v5 Events allow users to simultaneously participate in the same event instance.

There are two settings which can impact this:

1. Template *Permission* Level - If the template itself is locked down then the user may not be able to start, join or interact with it.
2. The *Location Rights* setting inside the Event template itself can further restrict participation by remote users.





The *Location Rights* block allows you to regulate how users can participate in running events.

- *Full Access* means that any user can join and participate in a running Event. They can click Manual Triggers, view the logs, see who else is participating, etc.
- *Read Only* means that any user can join and view the Event but they cannot interact with it. They will get updates as they occur and see who is participating but cannot click Manual Triggers.
- *Single Station* means that only 1 user at a time can join the Event. Once someone joins the Event then no one else can join (not even in *Read Only* mode) until that user leaves and gives up their slot. This does not prevent users from starting another instance of the Event on their own, it ensures that only 1 user is interacting with an Event instance at any given moment.

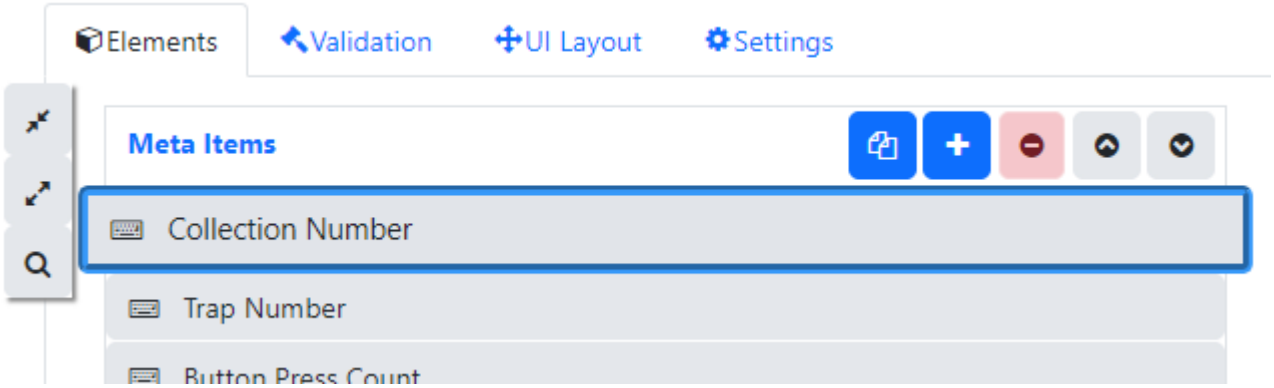
> The ability to view and start Event templates / data is restricted via the template permissions themselves (see (1) above).

Customizing Your Browser Title

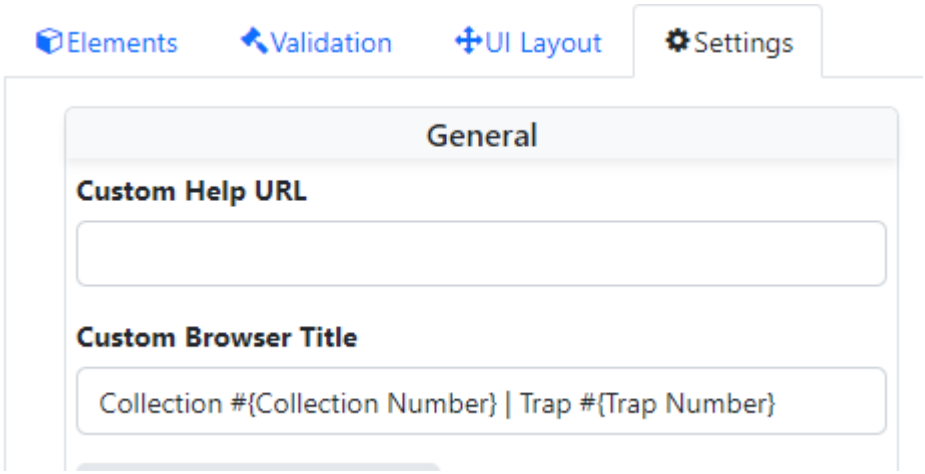
In the *Settings* tab you now have a Custom Browser Title text box. Leave this blank to use the default naming convention for your browser tab (includes the name of the event being run). If you enter anything in here it will override the default tab text with whatever you specify.

It might be confusing at first but you can also dynamically link Meta Item values into the title text by typing the name of the meta item you want inside curly brackets. For example, if you entered **Custom Title** - {NAME_OF_METAITEM_HERE} then the title would start with **Custom Title -** followed by whatever the current value is of the Meta Item named 'NAME_OF_METAITEM_HERE'.

So take a demo template with 2 Meta Items of interest (Collection Number and Trap Number)

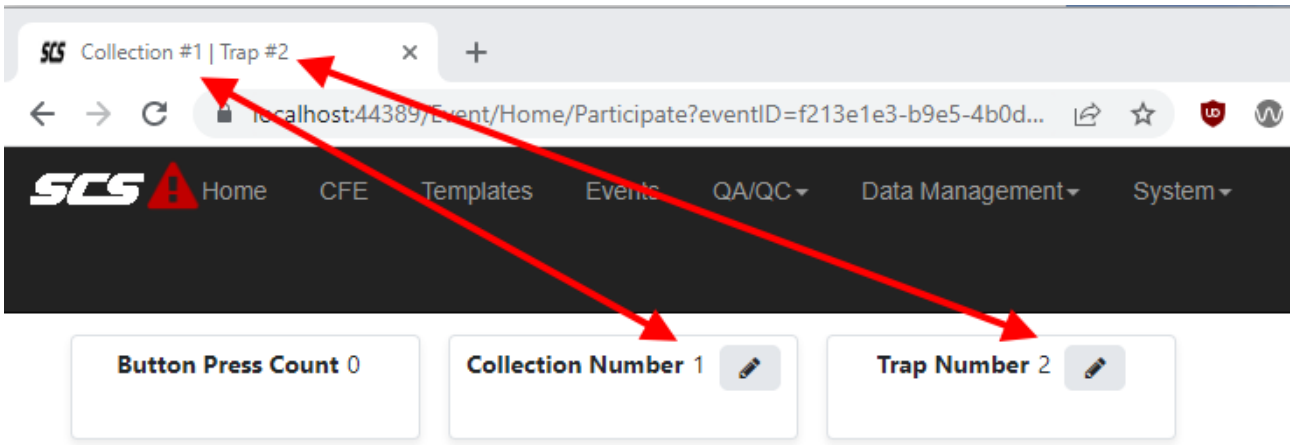


and a Custom Title of Collection #{Collection Number} | Trap #{Trap Number}



When the event is run you'll notice the title is changed to match your 'custom title' and all Meta Item names inside brackets are replaced by their current value. If a Meta Item of interest changes (e.g. someone clicks a button which makes the Trap Number increment to 3) then the title will automatically change on everyone's screens as well to reflect the new value.

So in our example, looking at *Collection #{Collection Number} | Trap #{Trap Number}* we have two Meta Item references - *{Collection Number}* and *{Trap Number}*. When the event is run, these two placeholders will be replaced with their respective values (SCS will determine the current value of the Meta Item named 'Collection Number' and replace the entire *{Collection Number}* text in the title with the Meta Item's current value, etc).



Event Template Editor - User Interface Layout Tab

UI Layout

The UI tab allows you to arrange the display your users will see when running the event. All relevant components with UI representation will be displayed on this tab so you can get a preview of what the participants will actually see when they are interacting with this template.

There are two primary modes:

1. *Auto Layout*

2. *Manual Layout*

To swap between Auto & Manual layouts simply check/un-check the *Auto Layout* checkbox in the *Layout Management* portion of the UI.

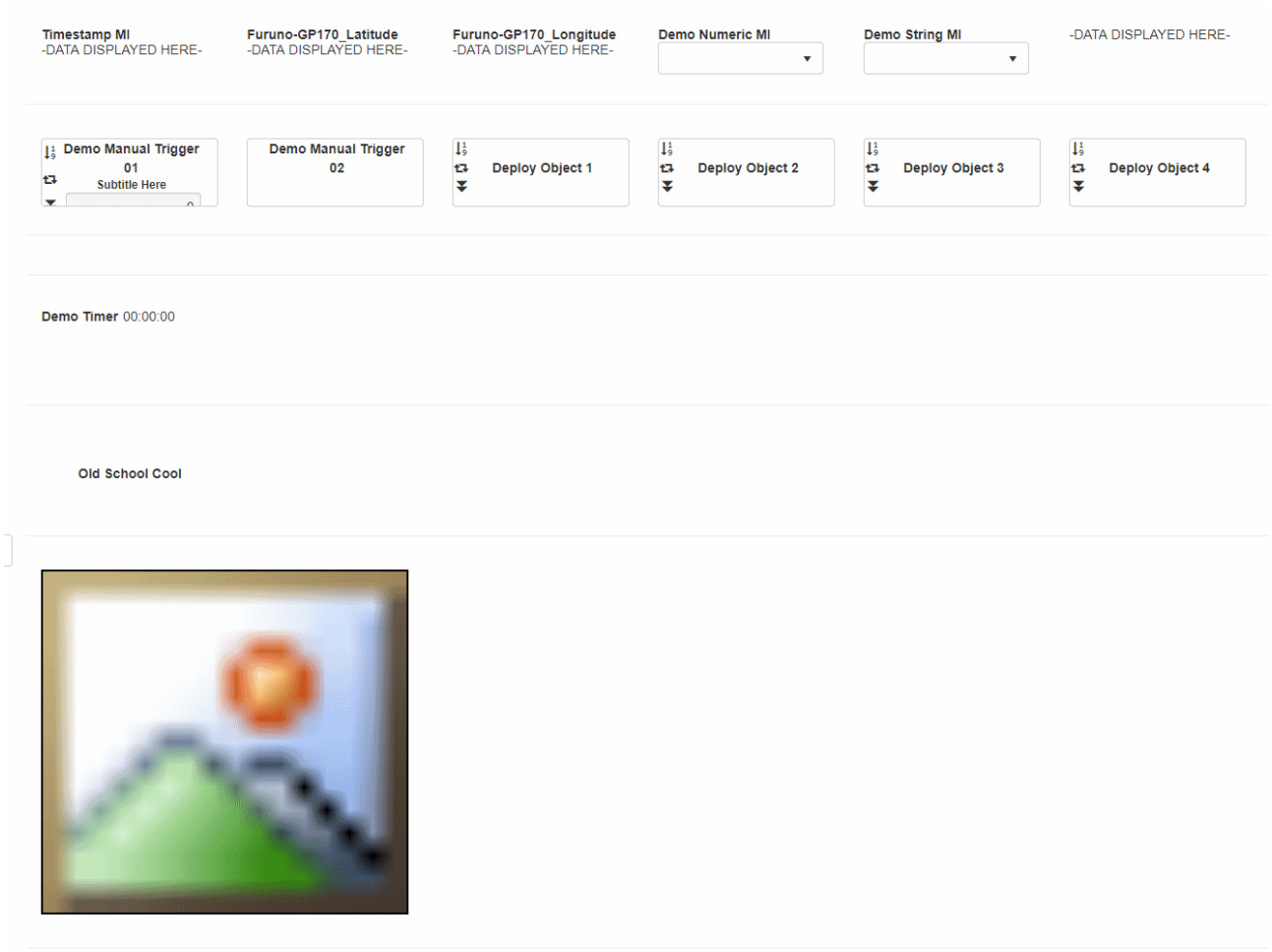
Auto Layout

The automatic layout will use a responsive web design (RWD) to display all your components in a linear fashion. This mode has two primary advantages. First it requires minimal work on your part, what you see is what you get. You don't have much control over the arrangement other than being able to resize some of the components. Secondly it will handle all the different screen resolutions users may run the event on, from a huge display in the lab to a small mobile device. The downside is you have no real control or say in how the layout is arranged. You cannot optimize the Event by grouping things together, resizing individual components and so forth.

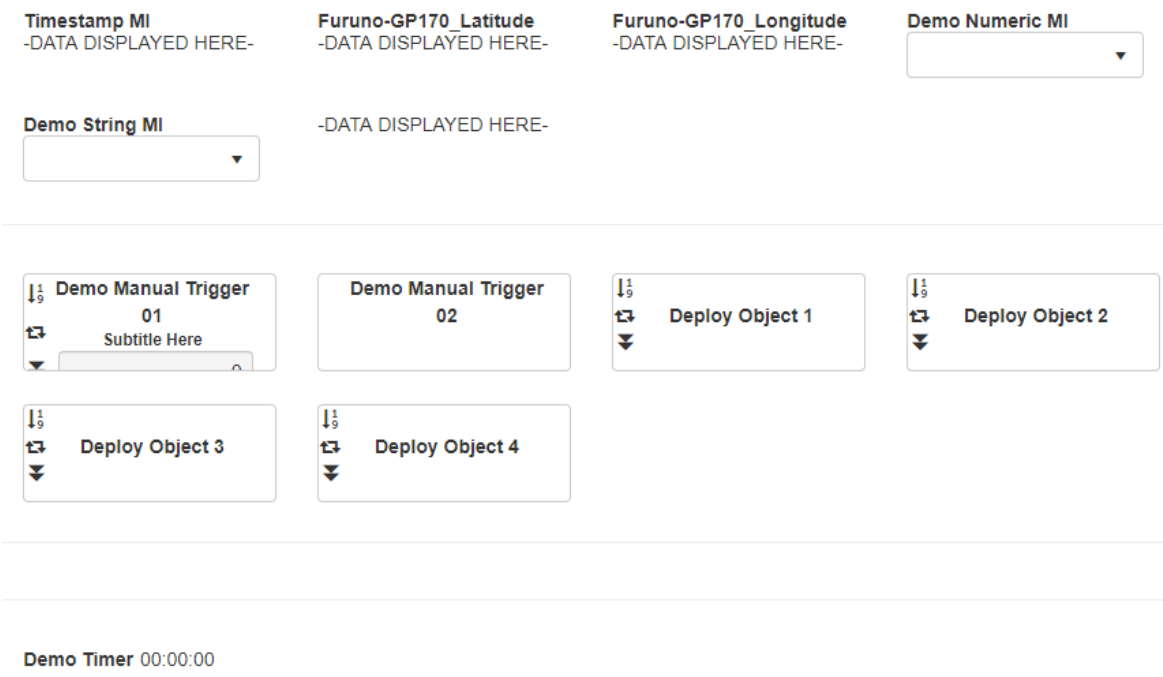
The *Auto Layout* works by laying out horizontal rows of component types. If there are more instances of a type then fit in the row then it wraps down to the next line and continues in this fashion until all instances of a given type have been displayed. At that point it moves on to the next component type. The order in which components are displayed is:

1. Meta Items
2. Manual Triggers
3. Progress Bars
4. Timers
5. Labels
6. Images

This results in a UI similar to the below, Meta Items are displayed in a row followed by Manual Triggers and so forth.



If the horizontal resolution of the screen were to shrink then the UI controls would wrap onto into a new row, as seen below.



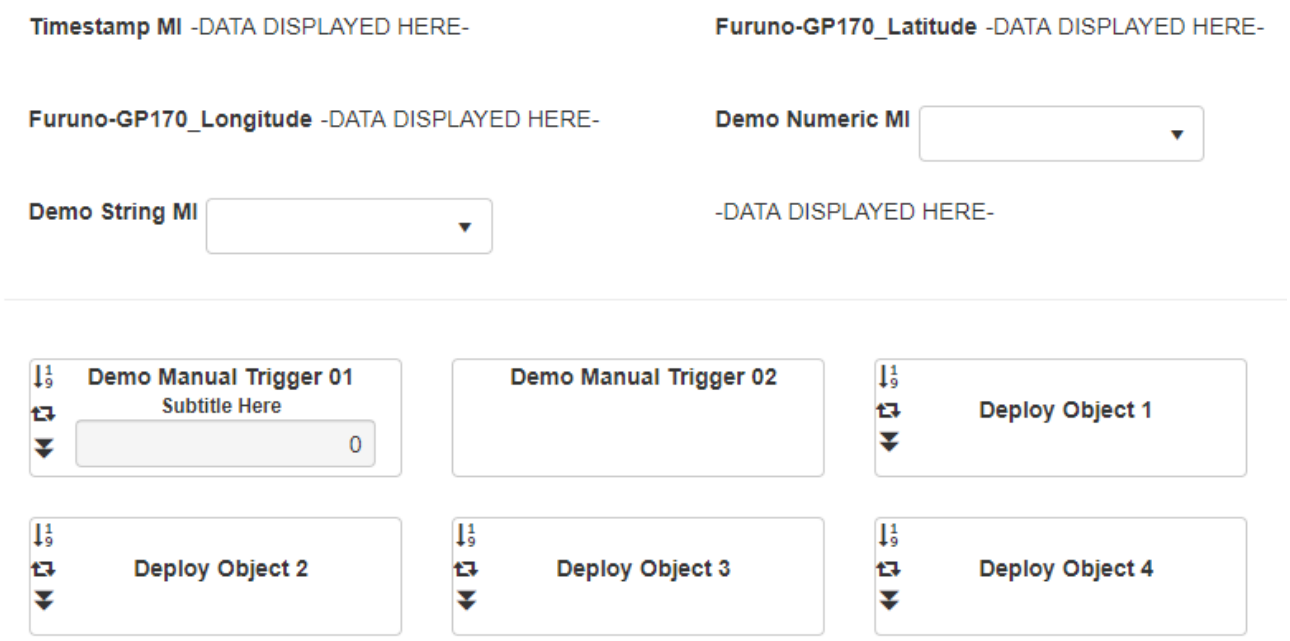
Note that 2 of the Meta Items and 2 of the Manual Triggers exceeded the horizontal width of the UI and thus were placed into a new row. The smaller your horizontal resolution the more rows you would potentially have.

> This may cause a vertical scroll bar to appear if you start to exceed your vertical resolution as well!

This is a very convenient way of displaying things, however it has it's issues. For instance many of the components are to small to support their contents. Taking a look at the *Demo Manual Trigger 01* Manual Trigger you can see the "01" part of its *Label* has wrapped as well. To resolve this SCS allows you to set a width and height for three of the major UI components when in *Auto Mode*.

Increasing the widths of Meta Items to 40 pixels and the width of Manual Triggers to 24 results in a much cleaner UI. Each Meta Item *Label* is aligned to the left of it's value, each Manual Trigger is displayed without internal wrapping and each *Label* is displayed where it's *Label Position* setting indicates it should be.

> Changes to the width and/or height values in *Auto Mode* effect all components of the given type. You cannot apply width/height values to a single item.

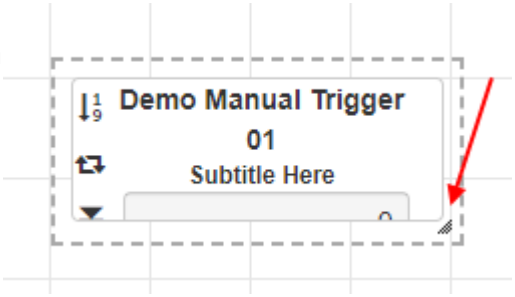


However, even though this looks better it may still not be ideal. Perhaps you want to put Manual Triggers next to the Meta Items that they update. Perhaps you don't want your Images or Labels down at the bottom of the screen but next to or inline with other related components. Perhaps you want one button to be larger than the others. For this and many other reasons Events also have a manual layout mode.

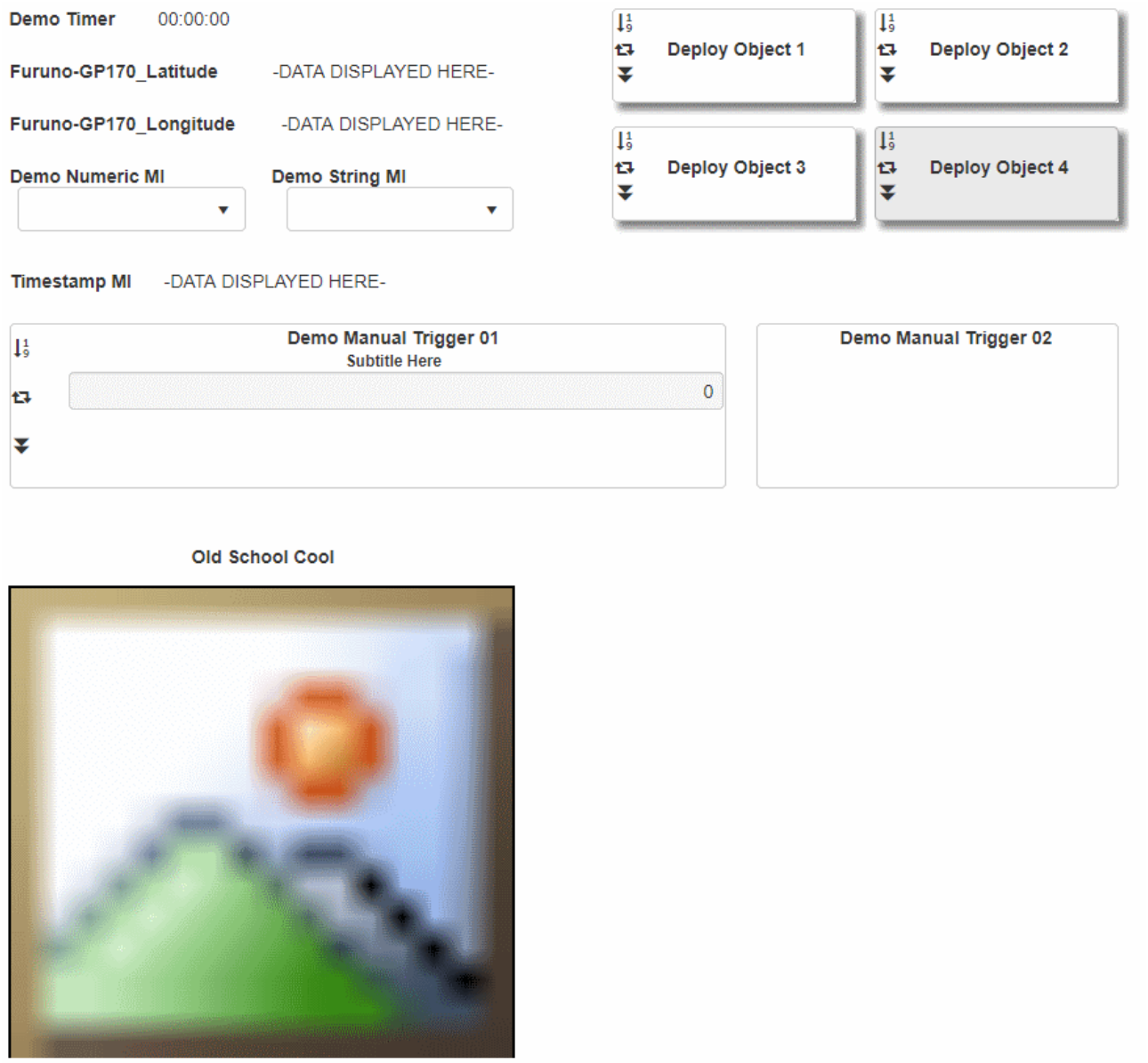
Manual Layout

The manual layout users a similar pattern as the primary SCS Layout page. You are presented with a grid upon which you can manually resize and move all components for your UI. You have fine grain control of how things are arranged on the screen, however this is not a responsive pattern and requires some initial involvement on your part to get it setup in a manner which makes sense.

When laying out in Manual Mode each element with a UI representation is added to the screen on top of a grid wrapped in a dotted box. You can use your mouse to click and drag the component anywhere inside the grid. In the lower right corner of each box is a handle which you can grab to resize the wrapper. Depending on the type of component you are resizing and it's settings (such as the *Labels Position* property) the component will resize and restructure to fill the space you have defined.



The *Manual Layout Mode* allows you to create a much more customized display, as seen below for instance. The downside is you have to lay it out manually so initially it takes a few moments to get things where you want them. It also saves the sizes and positions as hard numbers which disables the responsive aspects of the Auto Mode.



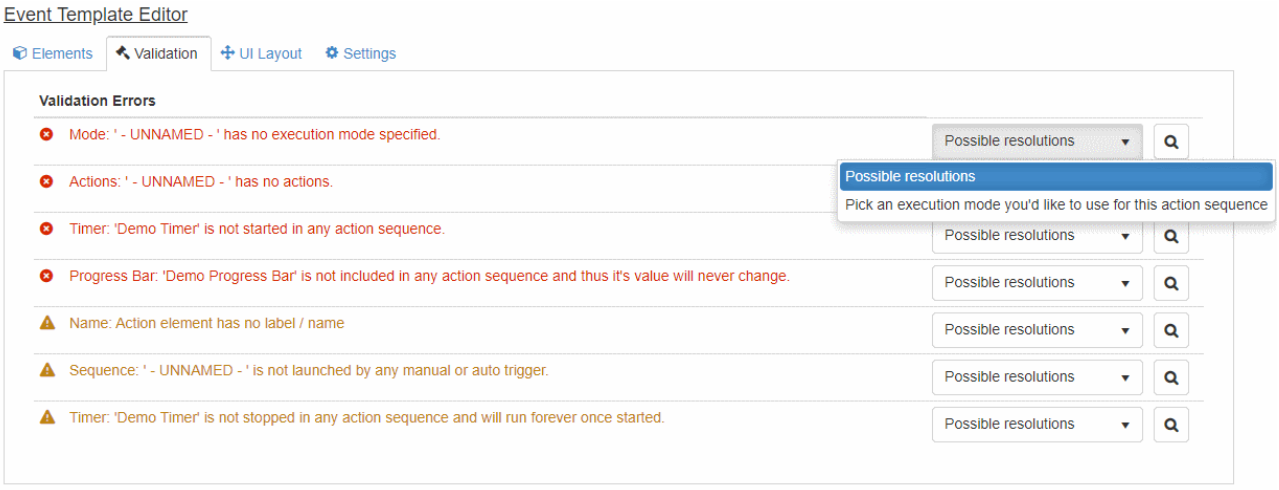
Event Template Editor - Validation Tab

Validation

As with other templates, the Event template must be validated prior to running. Attempting to *Save* or clicking the *Validate* button will run through all the validation checks for the loaded template.

If validation issues are found they will be displayed in a list categorized by their severity. At the end of each issue will be a dropdown list which will provide hints as to how to fix the issue. Clicking the Find button at the end of the row will attempt to automatically located the component and highlight it for you.

> ⚠ If the error is critical and prevents the template from serializing then your template cannot be saved until the issue is fixed. Errors will allow your template to save but not run. Warnings and below are recommendations but do not block you from moving forward if you choose to do so.

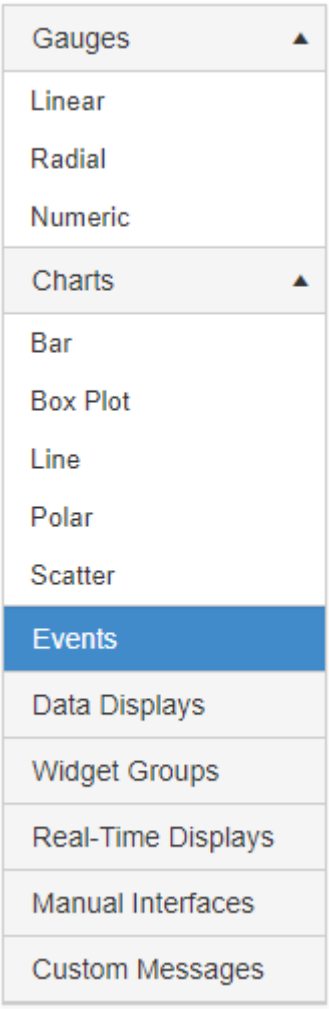


Event Template Editor

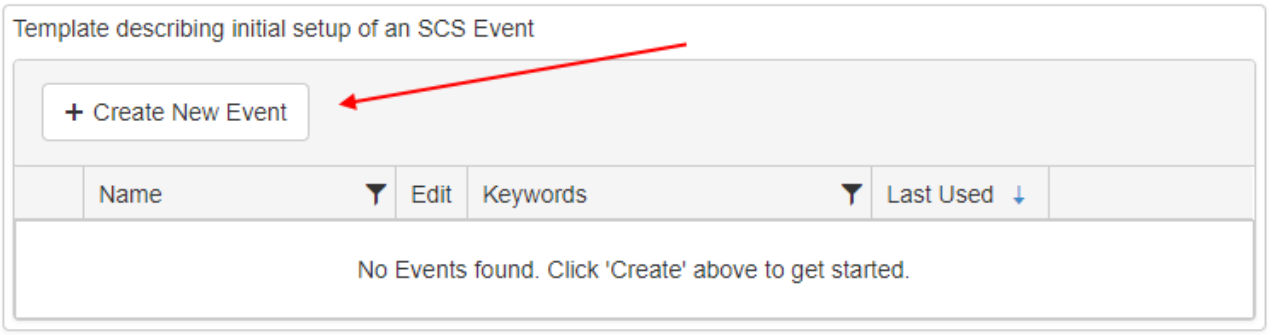
The *Event Template Editor* is accessible in the same manner as other templates via the Manage My Widgets portion of the website.

> Event templates are not treated the same as other widgets as they can not be included into any window layout. They can only be run via the Event Management portion of the site.

To get started:



1. Browse to the Manage My Widgets page
2. Click the ***Events*** item in the widget selection panel-bar
3. Click the *+Create New Event* button which is on top of the grid that appears listing all existing event templates



4. Fill out the popup window with appropriate values and click *OK*
5. After a blank event template is created you'll see the the standard template layout with the editor targeting Events

Editor Overview

The *Event Template Editor* essentially is a tab strip dividing up the work and feedback so you aren't overwhelmed with a huge UI. Each tab can be navigated to without worry about losing the work you have done in any other tab. However, as with other templates, you must click *Save* to make any of your changes permanent. Clicking between tabs does not save your work.

The tabs comprising the template editor are:

- Elements
- Validation
- UI Layout
- Settings

> ⚠ Changes made to existing event templates do NOT effect any prior or currently running events. They only take effect on the next run of the template!

As with other templates, the ability to *Save* and *Validate* can be found at the bottom of the screen via the corresponding buttons.

Event Management

SCS Events are defined via templates which are created and edited using the template builder like other templates. Once a template has been defined and passes validation checked it can be run. Events differ from most other templates here in that the Event UI cannot be added to your normal SCS `Layout`, instead it has to run in a dedicated full screen browser window or tab via the *Event management* page.

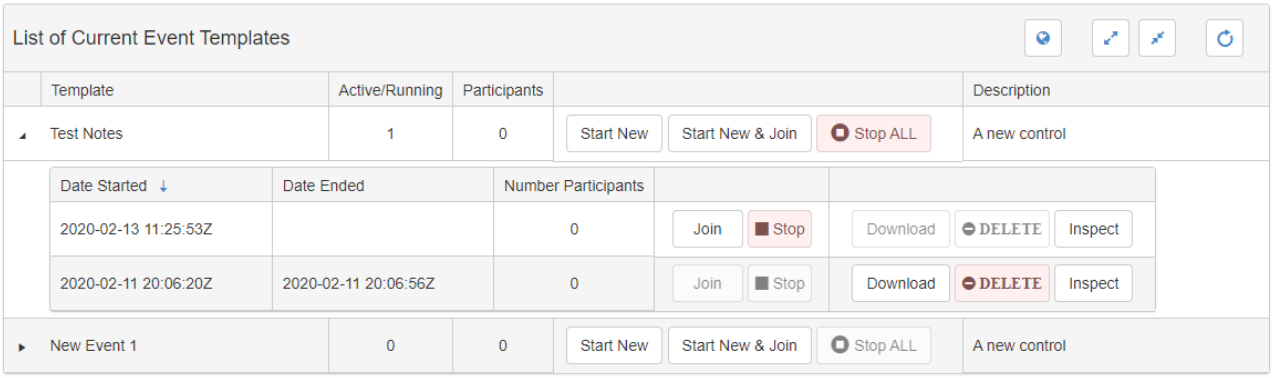
The management page has a single grid on it, within that grid is a list of all event templates that your [currently logged in] account has access to. The grid has multiple columns:

- The first column is the name of the template / event
- The second column tells you if any instance of that template are currently running
- The third column tells you how many users are actively joined to and participating in the active instance of the event
- The fourth column provides various commands allowing you to start a new instance of the event, start a new instance and immediately join/switch to it or kill all running instances.

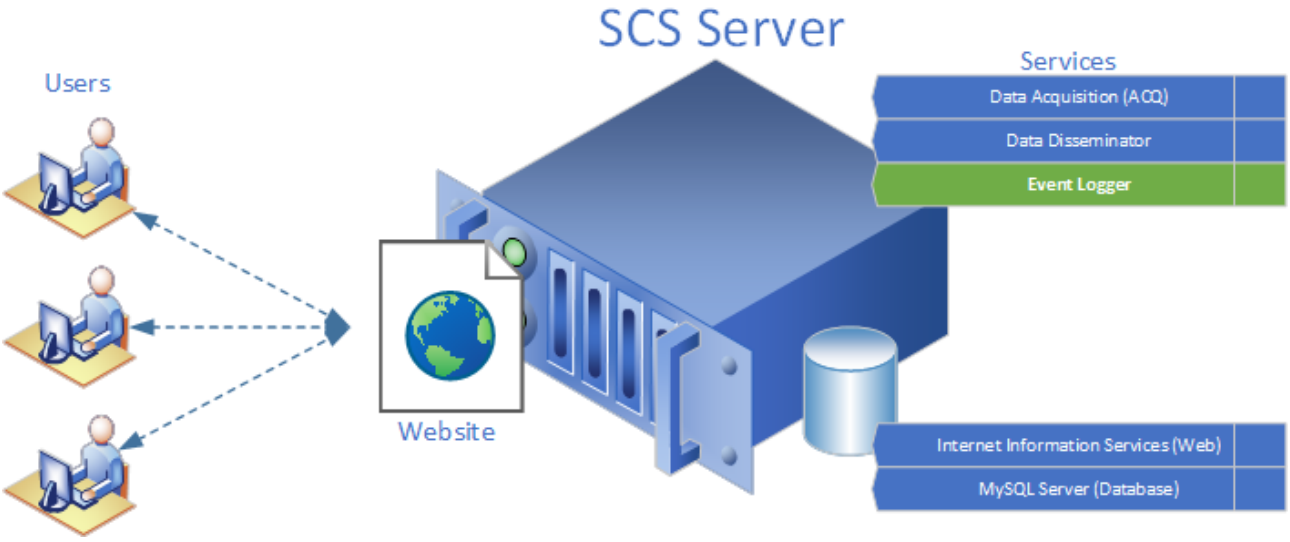
For each event template you can click the arrow to expand and see a details pane. This pane will let you manage the actual running or previously finished events. This is where you join an event which is already running, download or delete the data from prior runs and so forth.

> Joining or leaving (closing your window/tab) an event does not have any effect on it. If you leave it will continue to run in the background and you can rejoin it later. The only way to stop or end an event is via the appropriate `Action` or via the Stop buttons mentioned above (not recommended).

The grid has a toolbar which allows you to expand/contract all, refresh the entire grid and most importantly manage the Global Meta Items



The users interact with the Events via the website, however the actual process running all events is a windows service running in the background. The website only provides users with a point-and-click interface to this service. This service is why you can log in, interact with an event and log out without terminating the event. Unlike prior versions of SCS the event is NOT running on your machine under your account, it's server bound. It handles all the request from various users throughout the ship whether for a single event or multiple at one time and coordinates all the back and forth.



Global Meta Items

Global Meta Items (GMI) are Meta Items which can be referenced by all events but are centrally managed. A few key Meta Items are in almost all events, rather than change them all one by one you can reference a single global and take care of all references in one step. A good use case for these are things such as the current Cruise ID, who the current primary investigator is, what current fiscal year is, etc. Anything that might be useful for multiple events and should have the same value in each.

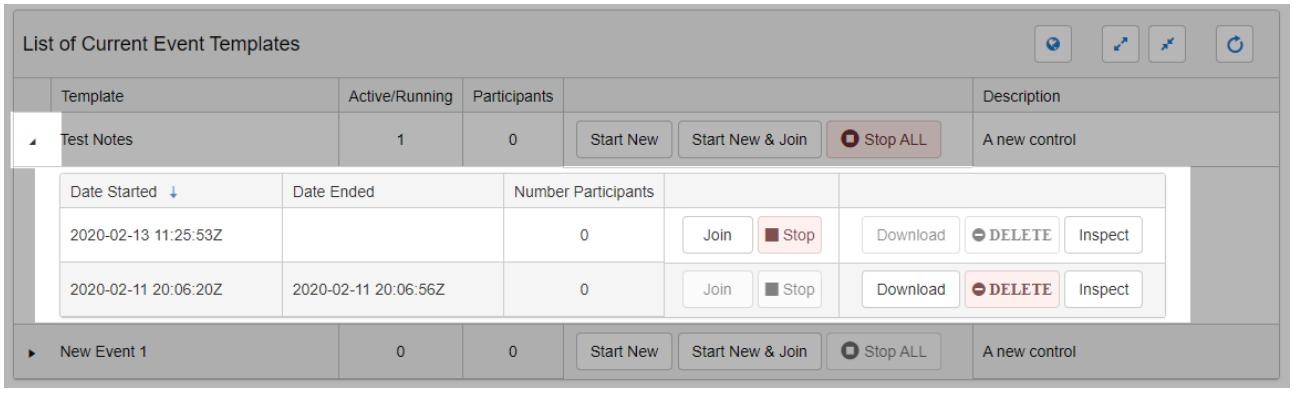
Grid Commands

There are 3 primary commands for each event template row.

- *Start New* allows you to create a new instance of the event based upon the last valid template.
- ***Start New & Join*** is the same as *Start New* however immediately after creating the new instance you are immediately joined to the event so you can start participating.
- *Stop ALL* immediately kills all running events of the given template type.

> ⚠ Stopping a running event via command buttons is not recommended (similar to killing a process instead of closing it and exiting gracefully).

Grid Details



Expanding an Event template will reveal all instances of that event which remain on the server.

If a value is present for *Date Ended* then the event is historical and you may download it's data via the *Download* command button at the end of its respective row.

If the *Join* button is enabled then the event is currently in progress, you may join it and potentially participate (based upon template rights and settings) by clicking the button.

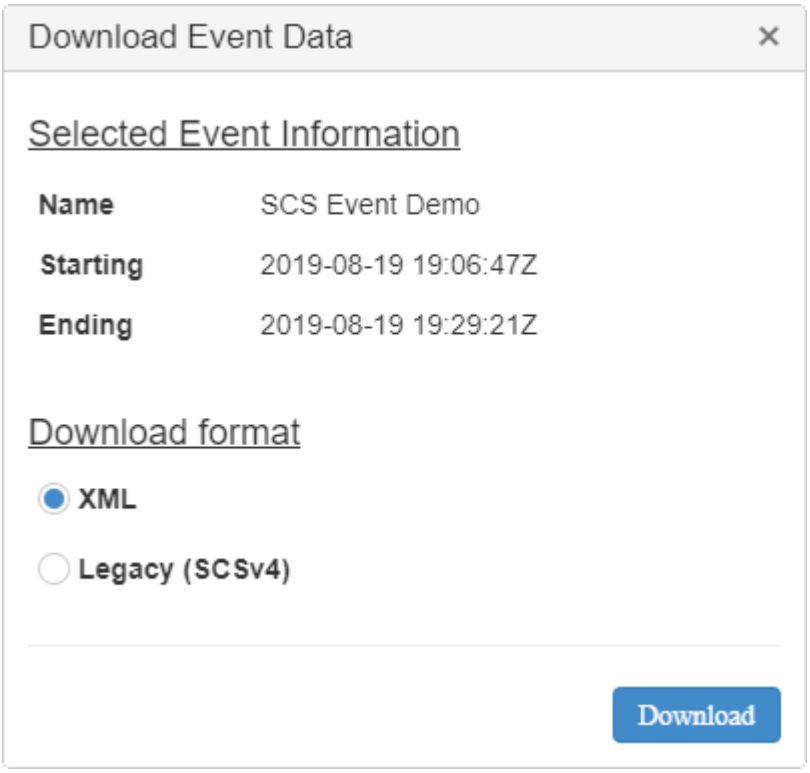
If the *Stop* button is enabled then the event is currently in progress and you may abort/kill it by clicking the button.

You can also *Delete* completed events and/or *Inspect* the data collected by both completed and running events.

Once you have created and/or Joined an event you will move on to the Event Participation screen.

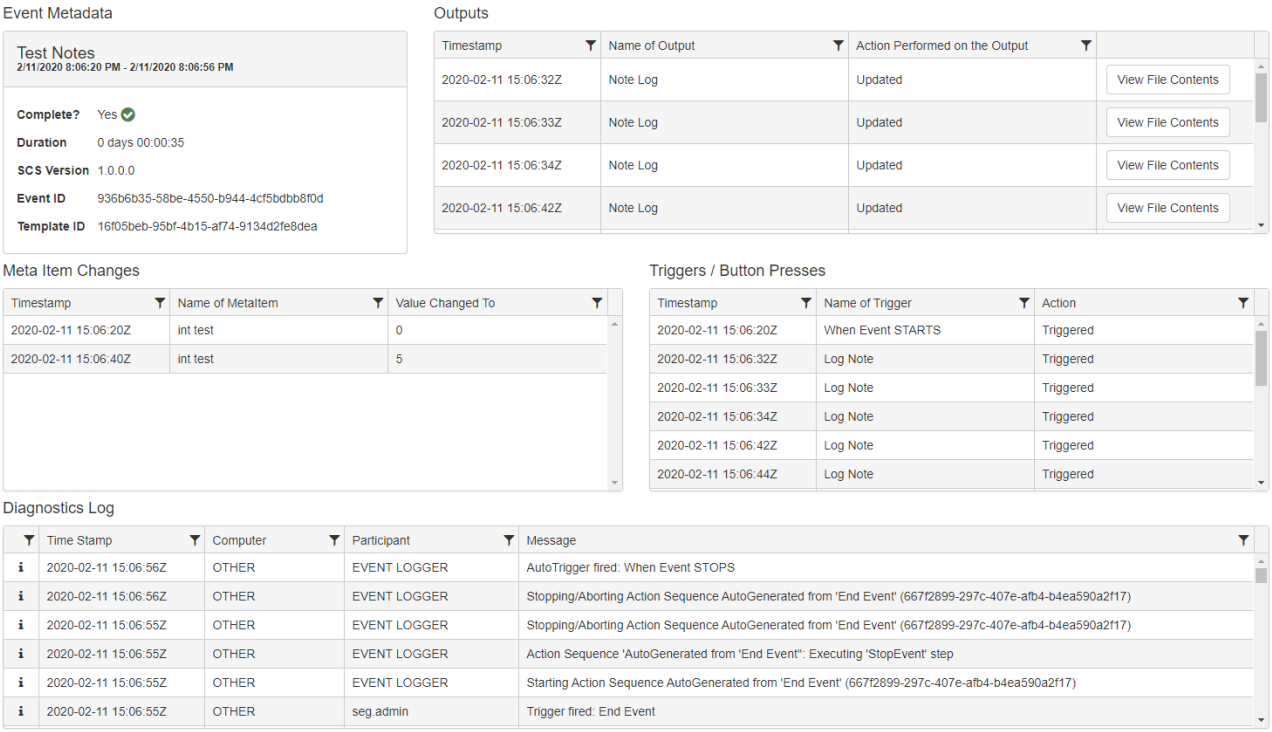
When downloading data for an event which has ended there are at least two formats you can choose from. If you are supporting legacy software you can choose the Legacy (SCSv4) option. This will attempt to

generate the same files and contents are prior versions of SCS (header files, elg files, etc). If not it's recommended you use the new XML format or NetCDF (if available) format for export.



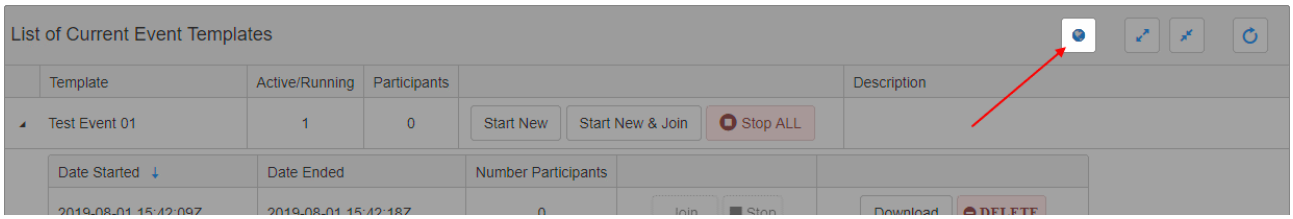
If you want to review the data collected without actually downloading it (or view the data collected during a live / still running event) you can click the Inspect button on the grid for the particular instance of interest. This will display a general overview of the event including changes to meta items, outputs, all button presses and the full diagnostics log.

This is just a summary and does not cover ALL data available in the XML download option, but it does give a good high level overview.

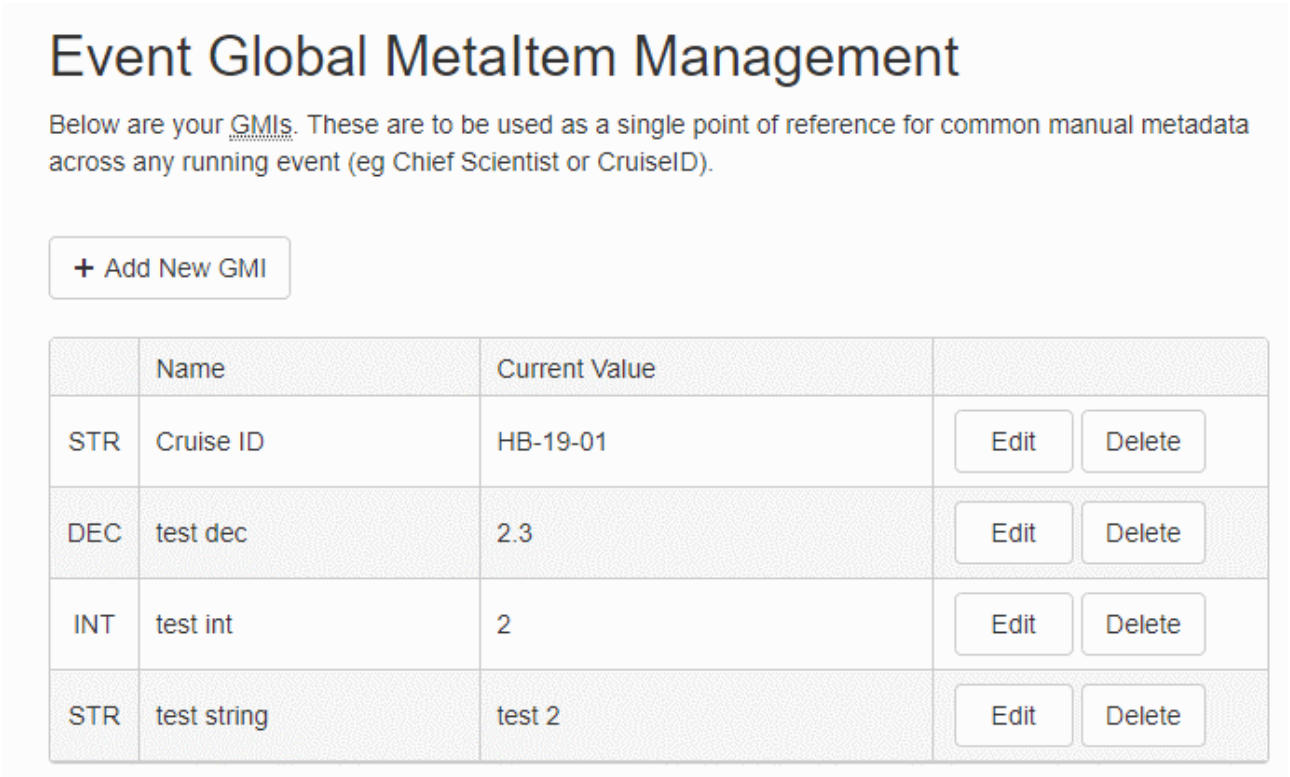


Global Meta Item Management

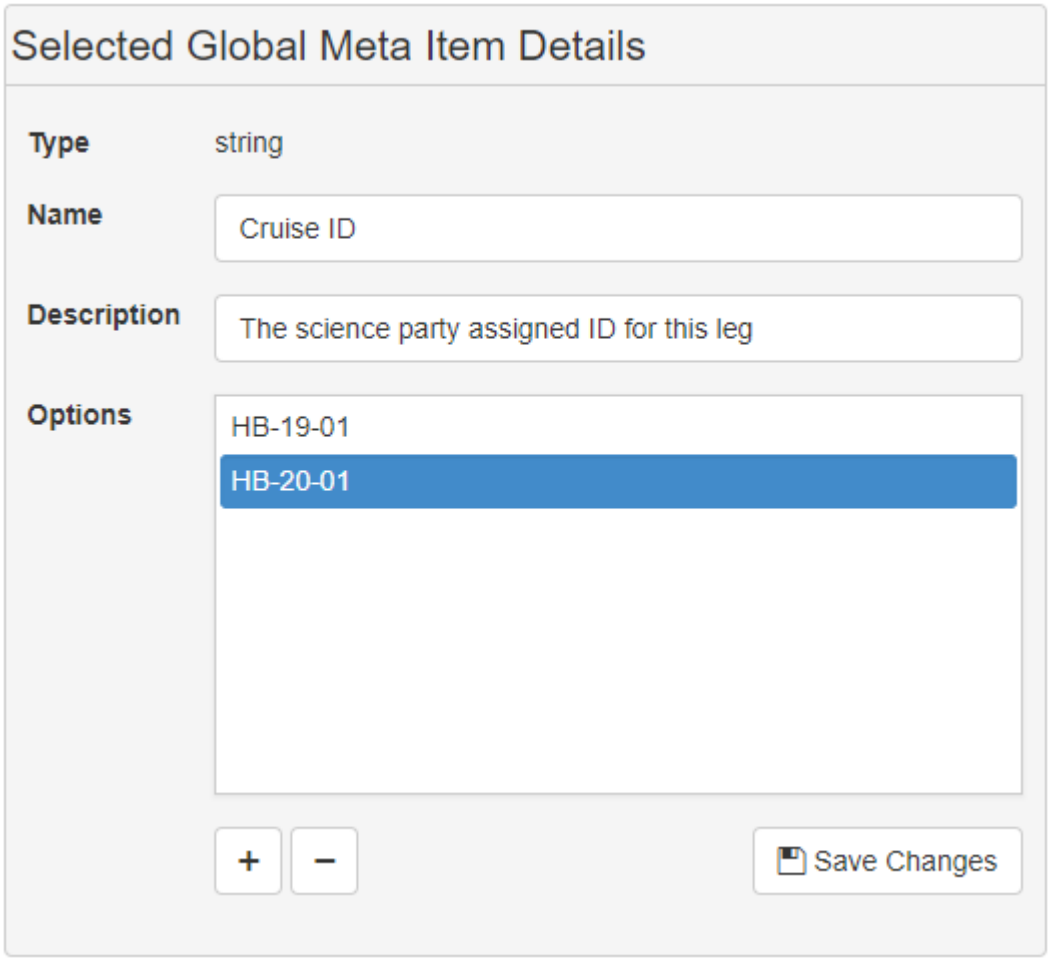
Creation and management of Global Meta Items is also done via this *Event Management* page. The UI to do this accessed via the toolbar of the main grid.



Once you click the globe icon you will be brought to the primary GMI screen. To create a new GMI click the *+Add New GMI* button. GMIs are not as advanced as the Meta Items you may be used to in Events. They can only be set to one of 3 types (string, integer, decimal), this type is decided upon creation and cannot be changed later. The main GMI grid is displayed below.



When creating/editing a GMI you can change the name, description and current value. Though you may specify multiple options only one can be active at any moment. Whichever value is highlighted (after hitting *Save Changes*) will be the value users will get when they update their references to your GMI in their events.



In the instance above the GMI named "Cruise ID" has a current value of "HB-20-01". Any event running which references this GMI will have it's value set to "HB-20-01" when it executes an Update Action.

> ⚠ It is advisable to update any and all GMIs prior to running your events (ideally prior to departure) to avoid incorrect values being saved into the them.

> ⚠ Do not forget to hit Save Changes to ensure your currently selected value is the one the GMI gets set to! Highlighting/clicking the value is not enough!

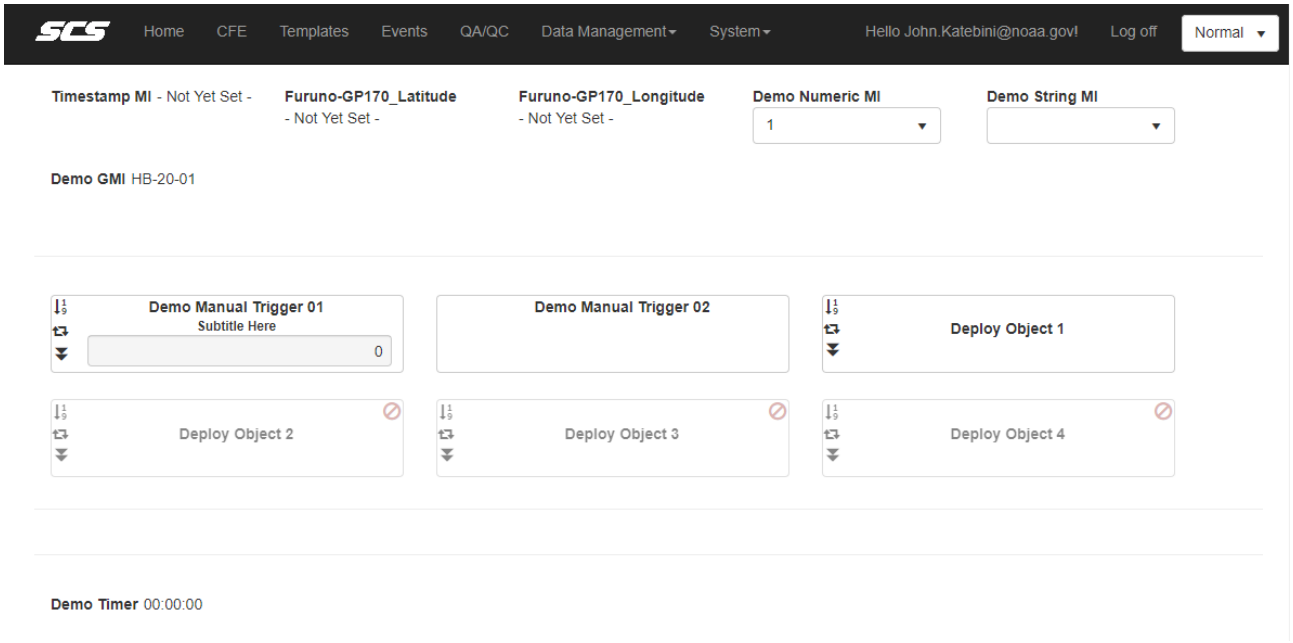
> ⚠ Be very careful when deleting GMIs. You will make any referencing event templates invalid if you do not clear them out of the template first!

Event Participation

Participation in SCS Events has changed drastically from prior versions. Depending on how the template permissions are setup and how the template *Location Rights* are set you may be able to join in on an existing running event with users from all over the ship allowing everyone to simultaneously fulfill their role in getting the job done together. Or, you can restrict access so that only one person may be in charge of the event at any given moment, everyone else can simply view (read only). Or you can lock everyone out completely and only one user can even see the event to begin with. The use case you have will help determine what rights you wish to delegate to others.

Upon joining an event, assuming you can view and participate, you will see a UI very similar to the one the template had been developed with in the template builder. Everyone else joined into the event will see the same thing you do, any change you make will be reflected on your screen AND theirs almost immediately.

This guide cannot provide guidance on how to run your event, or even what it will look like, as this is all custom/user defined. However details on how things should operate can be found in the template builder section.

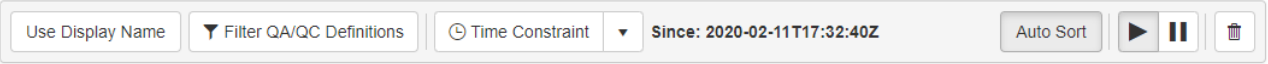


Quality Assurance / Quality Control Monitor

In the background the QA/QC service constantly skims records off the top of the database and performs tests defined by you against each row of data. The QA/QC Monitor page can be accessed by all (authentication is not required) and is used to review the state of your sensor suite based upon the rulesets you have defined inside the QA/QC Builder. It is actively updated as changes to QA/QC are detected and can/should be left open an monitored by human eyes for issues.

At the top of the page there is a toolbar which has a few useful functions on it. These options only apply to the specific tab you are browing on and do not effect the system at large or other peoples tabs/browsers. The functions allow you to:

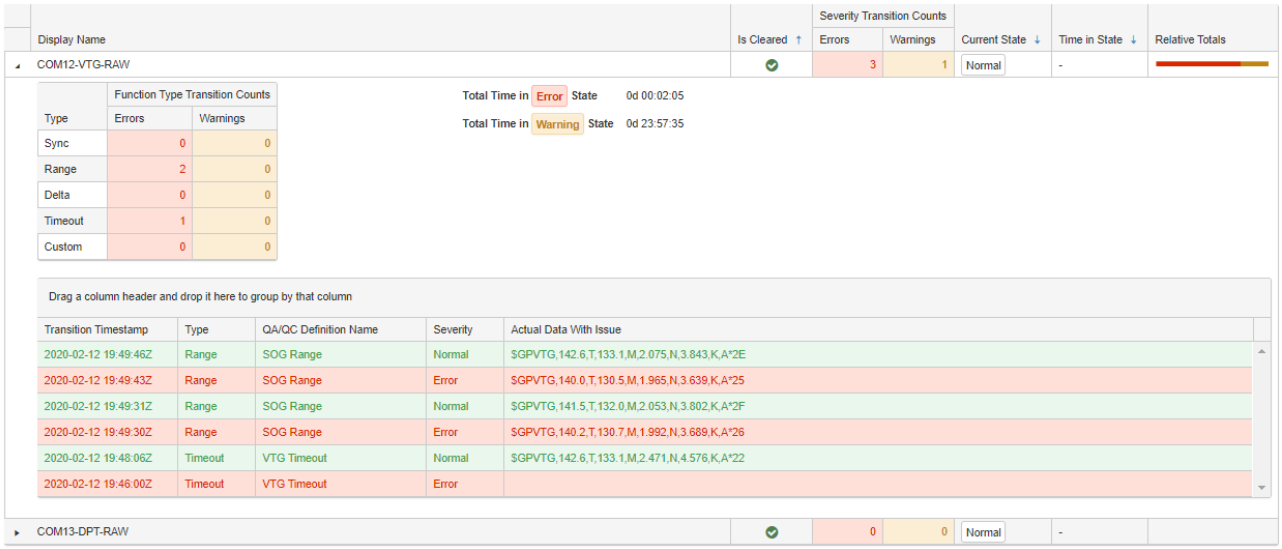
- Change the display to use system assigned naming vs user defined names (in CFE)
- Filter which QA/QC definitions are being reported
- Change the time frame the report is covering
- Automatically sort the display as data is updated to provide the best general feedback (you can change the sort yourself by clicking on a column otherwise)
- Pause incoming notifications so you can review the state statically
- Delete historic trigger records (authentication and appropriate rights are required to perform this action)



Below the toolbar is the main feedback portion of the QA/QC tooling in SCS. If auto-sort is enabled then the most pressing issues will show up on the top (most errors, currently in an error state, etc). However you can manually click on any column to reorder things as you see fit. To go back to the default either refresh the page or click the auto sort button in the tool bar.

The first column displays the name of the feed which has experienced an issue, the next column indicates whether it is currently continuing to experience issues or if it has been cleared. The following two columns display the number of times the feed has toggled between the various states (eg number of times it entered an Error state and/or a Warning state) followed by a column which shows which state it currently is in. The next column tells you how long it's been in it's current state. The final column shows you a relative graph detailing how many errors/warnings it's had compared to other items in your list.

Expanding the row by clicking the arrow on the left/start of it will provide details regarding those metrics. Here you can see the types of transitions which sum up to the above numbers and the totals in terms of time that the feed has had errors and warnings. You can also see each detailed message associated with the transitions, including the source data which triggered the change.



Display Name	Is Cleared	Severity Transition Counts: Errors	Severity Transition Counts: Warnings	Current State	Time in State	Relative Totals
COM12-VTG-RAW	✓	3	1	Normal	-	
COM13-DPT-RAW	✓	0	0	Normal	-	

Type	Function Type Transition Counts: Errors	Function Type Transition Counts: Warnings
Sync	0	0
Range	2	0
Delta	0	0
Timeout	1	0
Custom	0	0

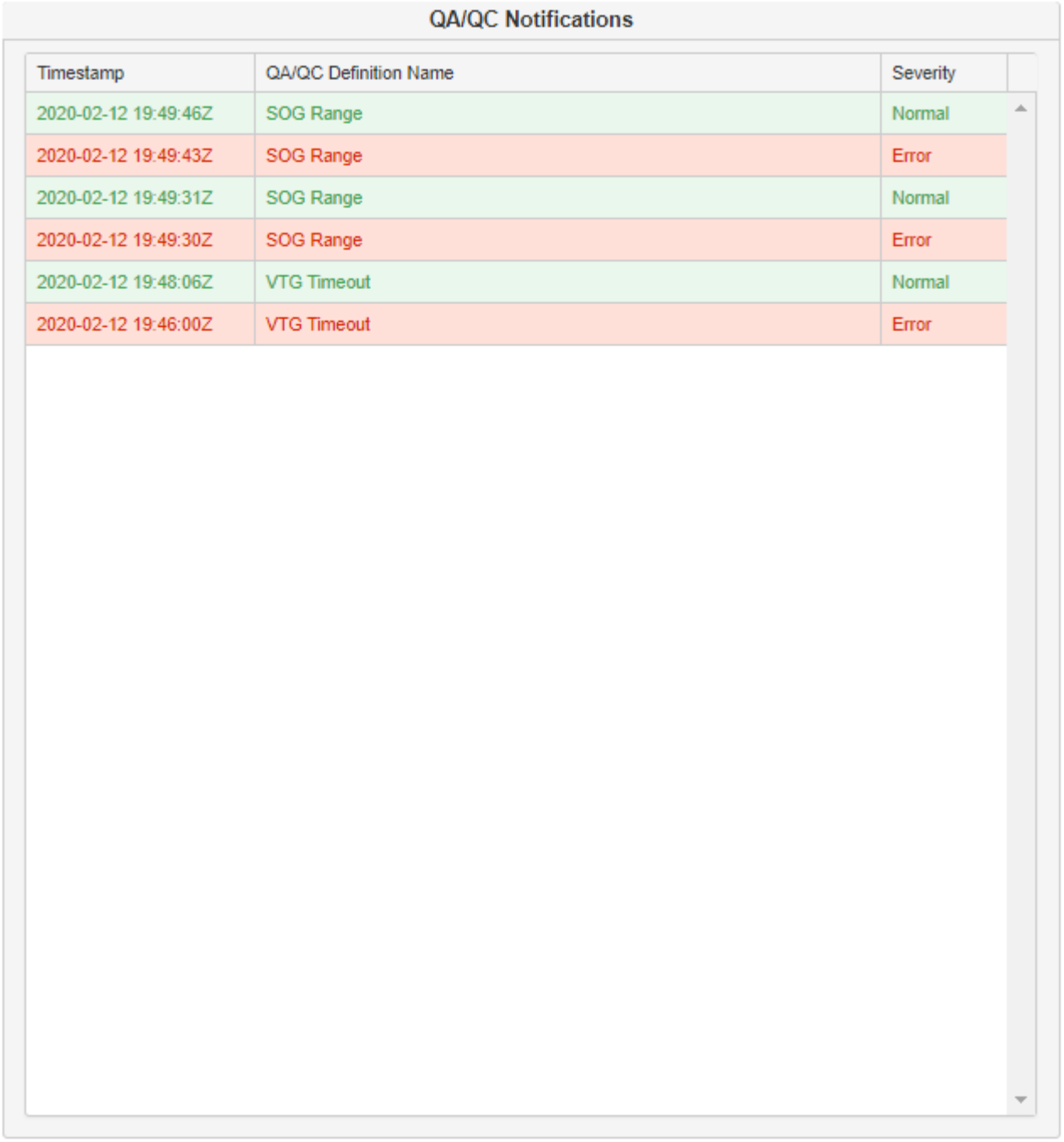
Total Time in Error State 0d 00:02:05

Total Time in Warning State 0d 23:57:35

Drag a column header and drop it here to group by that column

Transition Timestamp	Type	QA/QC Definition Name	Severity	Actual Data With Issue
2020-02-12 19:49:46Z	Range	SOG Range	Normal	$GPVTG,142.6,T,133.1,M,2.075,N,3.843,K,A*2E
2020-02-12 19:49:43Z	Range	SOG Range	Error	$GPVTG,140.0,T,130.5,M,1.965,N,3.639,K,A*25
2020-02-12 19:49:31Z	Range	SOG Range	Normal	$GPVTG,141.5,T,132.0,M,2.053,N,3.802,K,A*2F
2020-02-12 19:49:30Z	Range	SOG Range	Error	$GPVTG,140.2,T,130.7,M,1.992,N,3.689,K,A*26
2020-02-12 19:48:06Z	Timeout	VTG Timeout	Normal	$GPVTG,142.6,T,133.1,M,2.471,N,4.576,K,A*22
2020-02-12 19:46:00Z	Timeout	VTG Timeout	Error	

To the right of the grid is a live feed of QA/QC messages will be displayed. These notifications are what are organized and tallied to build the grid with the data mentioned above.



Quality Control Editor

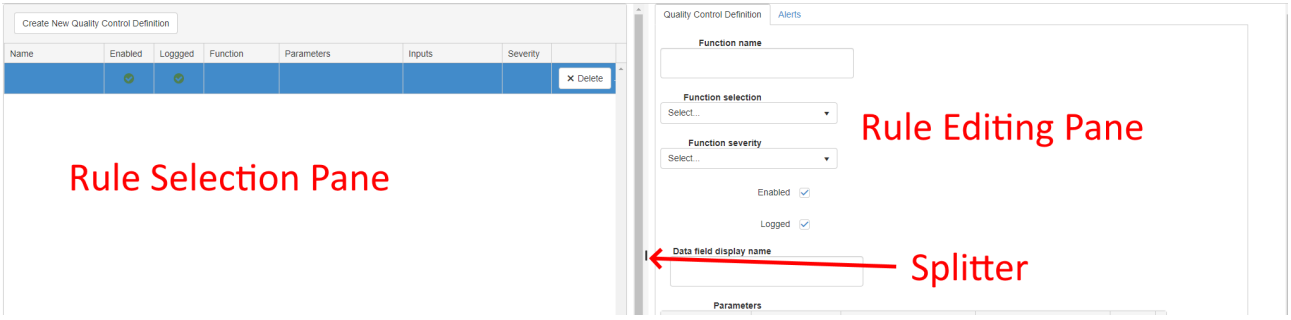
The Quality Control Editor page is accessible from the main menu under the QA/QC item. While this page falls under the purview of system administrators access can be granted to non-admin users via the *User Management* page by checking the *QA/QC Admin* right in the Additional Permissions section.

⚠ Be aware that any account granted the *QA/QC Admin* right will be able to create, edit and/or remove ANY definition as they see fit. It is recommended you only grant this right to responsible

> folks who have knowledge about what constitutes good QA/QC rather than giving free rein to any and all users.

The layout of the builder is similar to CFE with a list of selectable items on the left and an editing pane for the selected item on the right. In the middle is a splitter which you can use to change the size of either pane.

> When you save your changes they take effect immediately. No restart of any service is required.



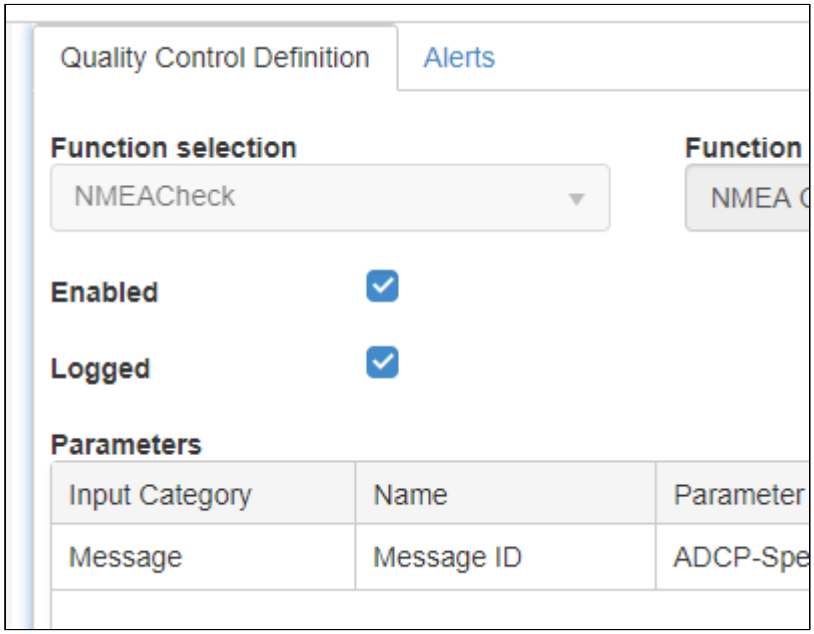
Defining QA/QC Rules

There are two main groupings when it comes to defining the rules QAQC utilizes to monitor and notify for quality: System-defined and User-Defined. You will notice the rule selection pane consists of two tabs which present the rules respectively.

System-defined QAQC | User-defined QAQC

Name	Enabled	Loggged	Function	Parameters
NMEA CHECK: ADCP-Speed-Log-Message	✓	✓	NMEACheck	• ADCP-Speed-Log-Message
NMEA CHECK: Barometer-RMY-Message	✓	✓	NMEACheck	• Barometer-RMY-Message
NMEA CHECK: CloudCover-Msg	✓	✓	NMEACheck	• CloudCover-Msg
NMEA CHECK: EK-120kHz-DBT-msg	✓	✓	NMEACheck	• EK-120kHz-DBT-msg
NMEA CHECK: EK-18kHz-DBT-msg	✓	✓	NMEACheck	• EK-18kHz-DBT-msg
NMEA CHECK: EK-200kHz-DBT-msg	✓	✓	NMEACheck	• EK-200kHz-DBT-msg
NMEA CHECK: EK-38kHz-DBT-msg	✓	✓	NMEACheck	• EK-38kHz-DBT-msg
NMEA CHECK: EK-70kHz-DBT-msg	✓	✓	NMEACheck	• EK-70kHz-DBT-msg
NMEA CHECK: ES-SDDBT-Bridge-Message	✓	✓	NMEACheck	• ES-SDDBT-Bridge-Message
NMEA CHECK: FS70-TrawlSonar-DBS-msg	✓	✓	NMEACheck	• FS70-TrawlSonar-DBS-msg

System Defined QAQC

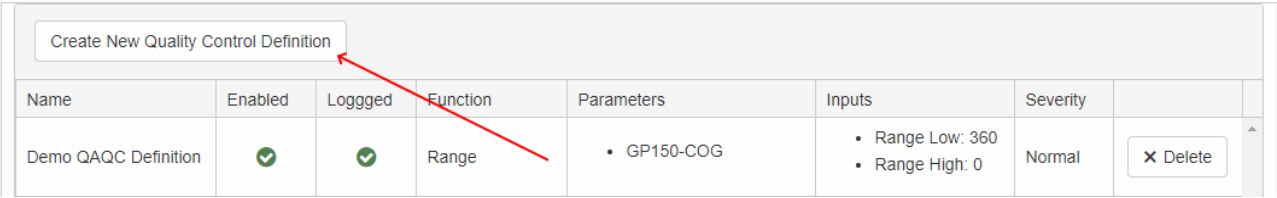


These rules are created automatically by the system to reflect the basic rule sets applicable to your current configuration. As you modify your sensor suite in CFE and publish your changes SCS will attempt to redefine these rules to accommodate your changes as well.

Since these rules are created and managed by the system itself you cannot tailor them to your needs. However, should one of them prove to be incorrect you can disable them or prevent them from logging so at least they will be ignored.

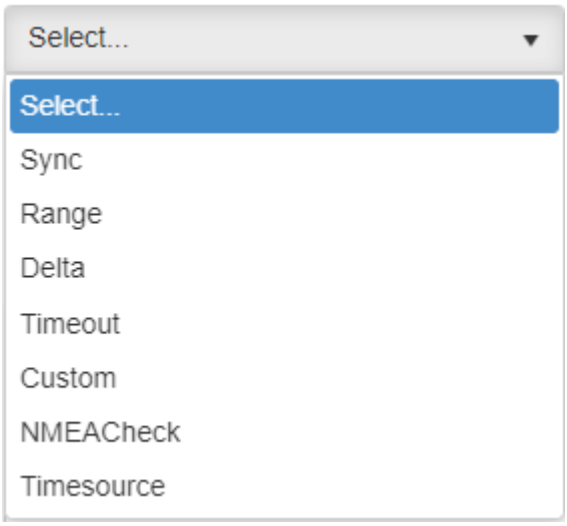
User Defined QAQC

Building out your ship's set of rules for monitoring the quality of the incoming data feeds starts by clicking the *Create New Quality Control Definition* button.



Once this button is clicked a new empty definition will appear in the selection pane and you can start defining it on the right in the editing pane. The editing pane consists of two tabs, the first is where you define what triggers the definition and the second tab defines how you want SCS to alert users should the rule be triggered.

Definitions may act on 1 or more sources, so name them appropriately. You will most likely have a set of standard rules for standard data types (e.g. below is a "Range" rule for a heading to ensure it is noted should it report outside a normal compass range) but you may also have advanced rules (for instance comparing all the GYROs to ensure they don't deviate from each other by more than 'x' degrees).



Many of the familiar function types from SCSv4 (DataMon) have transitioned into SCSv5.

Sync: QA/QC attempts to locate a Sync String in every message starting at the position given by String Position

Range: QA/QC verifies that a data field's value falls within the High/Low numbers you provide and trigger if the value falls outside this range.

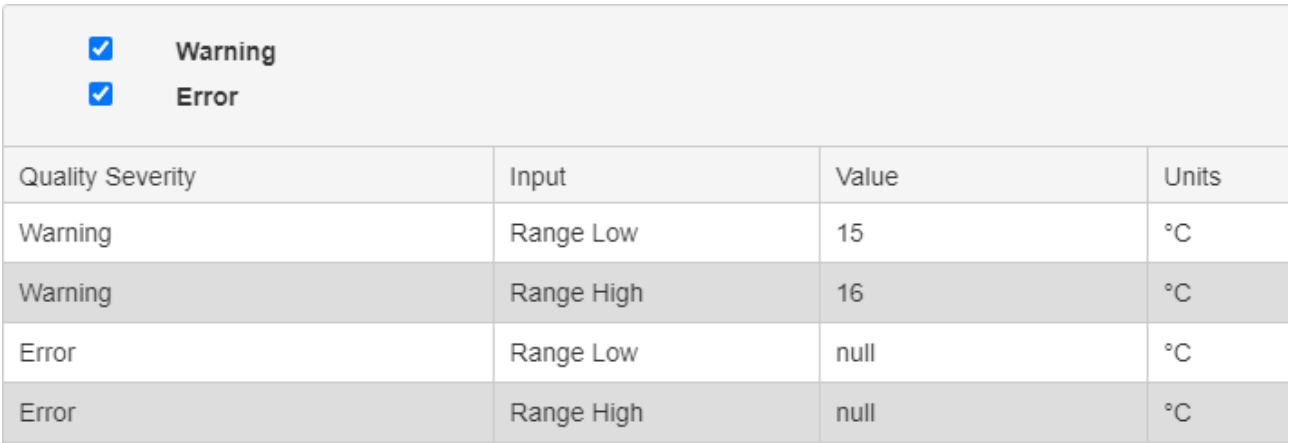
Delta: QA/QC compares consecutive values for a given data field and verifies that the absolute value of the difference falls within the High/Low numbers you provide. It will trigger if the value falls outside this range.

Timeout: Number of seconds to pass since the previous message before SCS considers a sensor's data to be "stale" and triggers the alert.

Custom: Users can also define custom equations, similar to the custom equations of Calculated Interfaces in CFE, which allow you to create much more advanced rules based upon your own logic (See also Custom Equations). The primary difference is that the resultant value is Boolean, a logical expression that returns either true or false. Similarly to CFE, users must assign data fields as variables and give them a friendly name to use in an expression. The timeout duration of a timed-out sensor is also available as an input category. See Custom Definitions for more information.

NMEACheck: NMEA Checks evaluate the checksum value of the NMEA message and the correctness of the message start and message label.

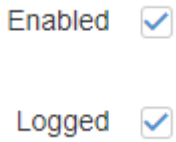
Timesource: Comparison against other systems providing official time (NTP, GNSS, etc).



For pre-programmed function, the user must also specify the severity of the rule for each relevant parameter. In the event this definition should trigger (e.g. fall outside the range you specify, etc.) then the issue could be construed as a Warning for your users and dataset or could be an Error. For instance, having a speed that's high or approaching a bad state but is still within reason might be a Warning whereas having a speed that's illogical (like a ship moving along at 50 knots) is most likely an Error.

Simply check the box to specify a severity range and then click the appropriate box in the *Value* column to set it.

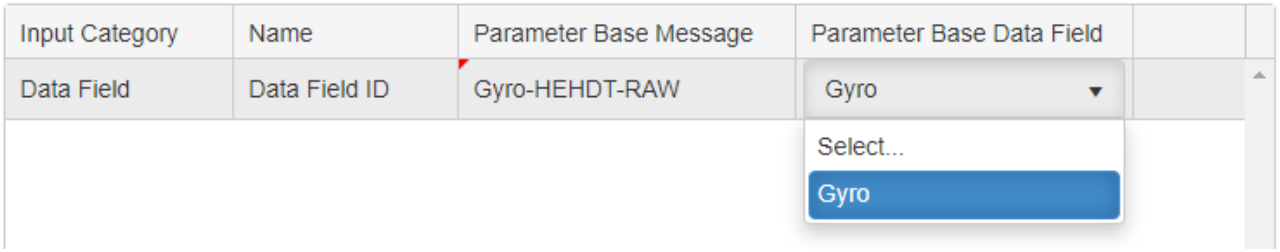
> A *Warning* state is not required before an *Error*. Some situations are binary and are simply right or wrong. For instance, if a GYRO is between 0 and 360 then things might be fine, but the instant it reads 361 that is an *Error*.



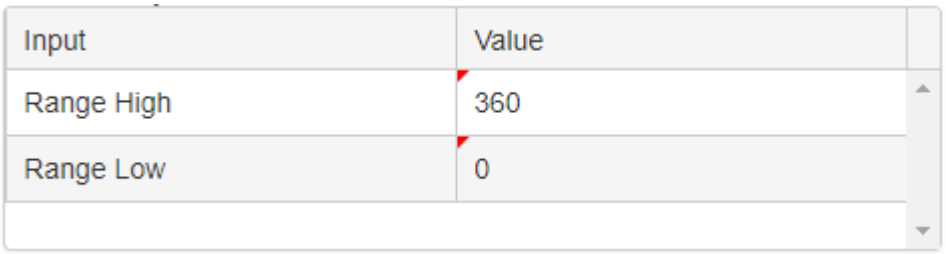
You can create as many QA/QC definitions as you wish. Generally the more the better as they provide in-situ feedback regarding the state of your sensor feeds. However, you may have rules you don't want to run at any given moment or you may have rules that are 'work in progress' or you may have rules which are triggering off a bad sensor you already know about and are slated to replace soon. Rules in these categories can be disabled by unchecking the Enabled box. Disabled rules will remain in the system but will not be evaluated and will be ignored until they are re-enabled. When a rule's state changes it is logged for future reference. You can disable this as well.

> ⚠ While it is tempting, it is not advisable to disable rules because they are annoying you. A better solution would be to fix the rule (if the logic is wrong) or fix the source of the problem it's alerting you to.

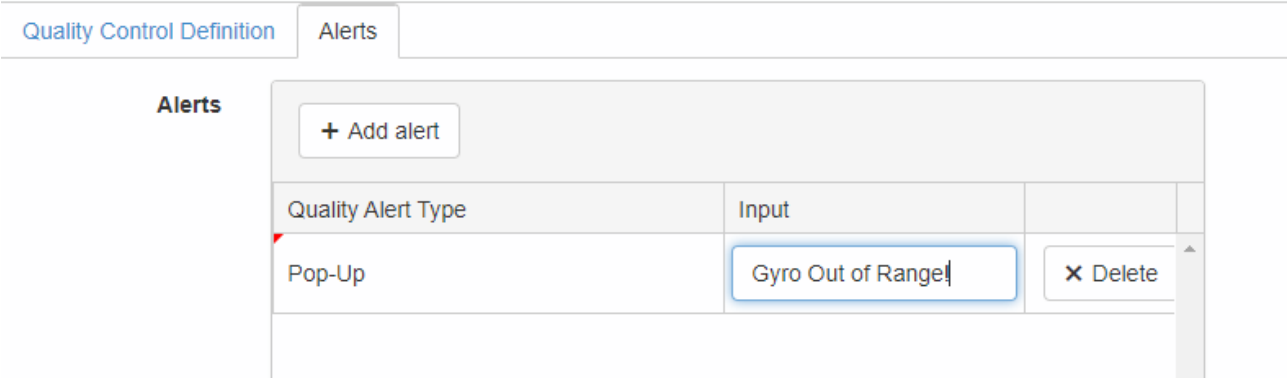
QA/QC Definitions work off sensor data, so depending on which function type you select you will have to enter one or more sources for which the rule will evaluate against. These data sources will appear in the Parameters section of the editor pane. You will be presented with drop down lists to choose what values you want to use for relevant row/columns.



Finally you must supply the logic which defines the rule. These are described by the *Inputs* section. In the instance of a *Range* check you would have to supply the High and Low values, if you were defining a *Timeout* you would supply the number of seconds before timing out and so forth.

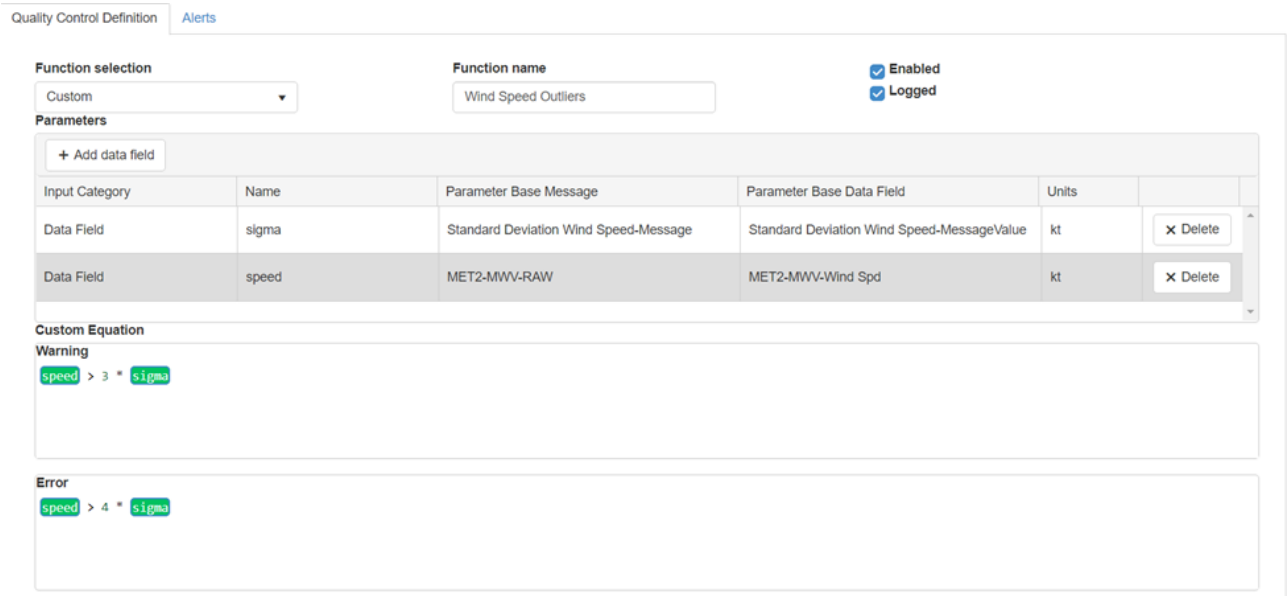


If you wish to add any alerts to your definition you may do so on the Alerts tab. Here you can specify ways of notifying people should your definition be triggered. This does not have any impact on whether it is recorded or noted in the QA/QC monitor and is simply an additional (active vs passive) means of notification.

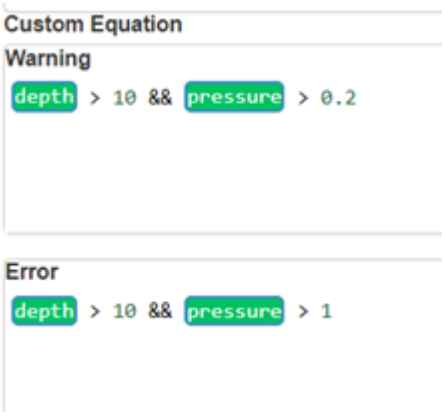


Custom Definitions

The custom logic equation in QA/QC allows for a Warning resultant severity or an Error resultant severity. If the result for the Error expression is false, then the Warning expression is evaluated. If the result for the Warning expression is false, then the resultant severity is Normal. Below is a simple custom equation that uses the result of the derived value of a standard deviation calculation for wind speed value that would be defined in CFE. In this example, if the wind speed is greater than four standard deviations, the severity is Error; if it is greater than three standard deviations, the severity is Warning; otherwise, the severity is Normal.



As with custom equations in CFE, the algebraic expression entered is what would appear on the right-hand side of an equation's equals sign (=). Given the friendly name variables and the provided functions and constants, the expression should otherwise follow the syntax the C# programming language. The conditional logical operations available are && for *AND*, and || for *OR*. For example, one could construct an expression using the friendly names *pressure* and *depth*



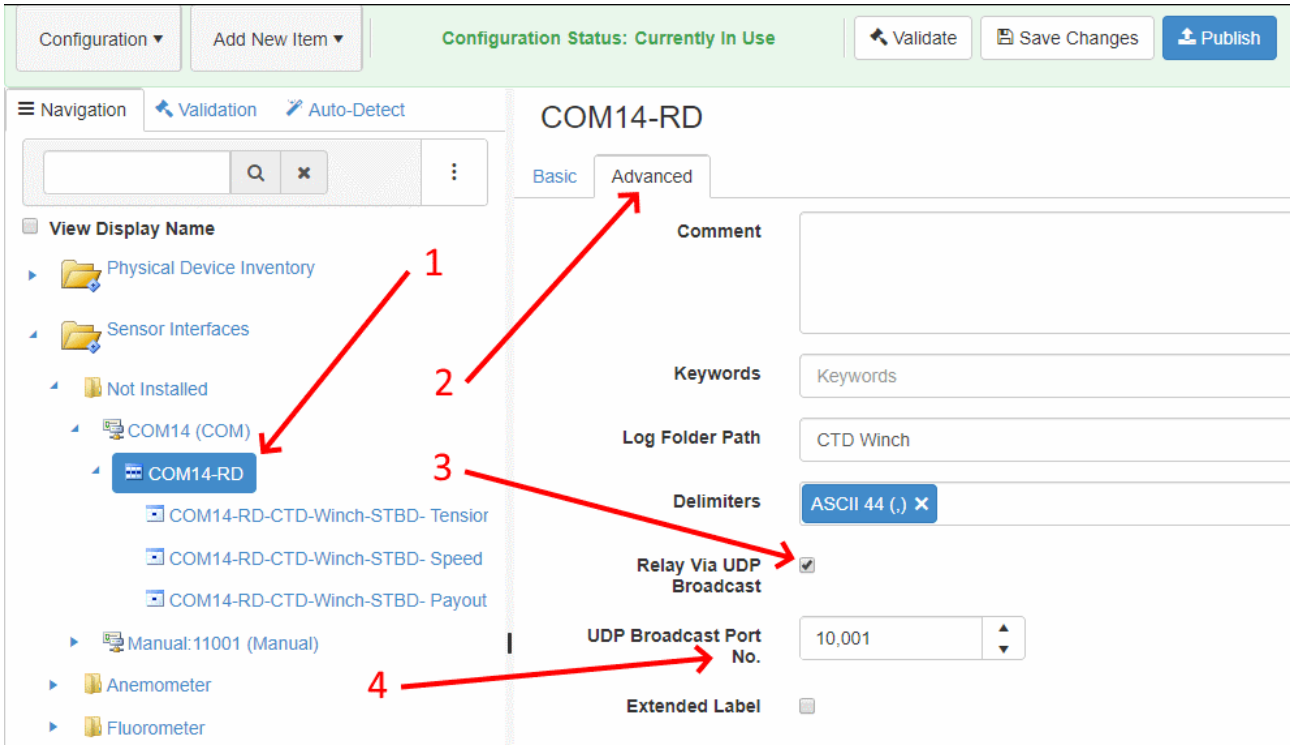
For custom equations, the user can program the Warning logic, the Error logic, or both. If either the Warning logic or the Error logic is undesired, simply leave blank.

Automated UDP Broadcasts

SCS now allows for real-time distribution of incoming sensor data to potential clients located on the ship's network via UDP/IP. When a line of sensor data is read by the Acquisition service if it is tagged for UDP Broadcast ACQ will send it out onto the network at the same moment it is written to the other output targets such as the database, RAW files, etc. Tagging a data stream for UDP broadcast is accomplished inside CFE.

The first step in tagging a data stream is to open CFE and select the Message Definition you wish to broadcast.

> Only Message Definitions can be sent over UDP using this method, if you want to send specific Data Field Definitions you must use a Custom Message.



Once you have selected a Message Definition, click on it's *Advanced* tab in the editor.

To send it's incoming sensor data out over UDP simply click the checkbox next to *Relay Via UDP Broadcast* and then enter the *UDP Broadcast Port* number you want it sent out on.

When your changes are finalized and you publish them ACQ will automatically start sending any data from this Message Definition out over UDP.

To view the UDP broadcasts as they are being sent out you may either listen on the port using a UDP client (outside scope of this document) or open the *UDP Broadcasts* page located under *Data Management* in the main menu. Each Message Definition you have enabled for broadcast will show up in the grid on this page along with the latest value sent over the network.

Automated UDP Broadcasts

Name	Port	Timestamp	Value
CTD-Winch-STBD	10001		

Note: *Automated UDP broadcasts are setup using the CFE component of SCS. If you wish to setup a custom UDP message or utilize multicast then you must do so via a Custom Message.*

If clients wanting the data are in a different subnet on your network you may have to modify your [managed] switches configurations to allow UDP broadcasts across the boundaries.

⚠ Modification of switching configurations should not be done unless you know what you are doing. Be careful not to enable broadcasts off the ship or you may flood your satellite connection with useless data and bog down your bandwidth.

Custom Messaging

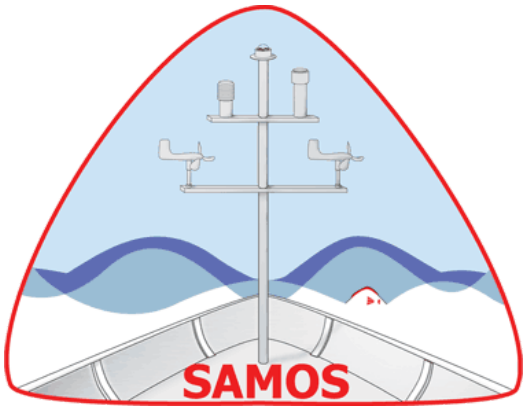
Custom Messaging is detailed in the *Widgets* section of the help.

See here for more information - Custom Messages

SAMOS

SAMOS (Shipboard Automated Meteorological and Oceanographic System) is a program operated by the Florida State University Center for Ocean-Atmospheric Prediction Studies (COAPS) to receive meteorological and oceanographic data from ships at sea in near real time, to verify the quality of that data and to provide feedback on the data quality to the ship. Details and further information can be found on the SAMOS website, including what data is covered, how to go about including your ship in the program, points of contact, submission formats and much more.

This help page only touches the surface and primarily covers how SCS interacts with SAMOS. Lower level details have intentionally been left out as they are the domain of SAMOS/FSU. It is highly recommended you visit the above links to learn more.



NOAA OMAO agreed to participate in this effort beginning with the NOAA Ship HENRY BIGELOW in the fall of 2006. Other ships were added over time and currently the entire NOAA fleet should be submitting their data to FSU on a nightly basis whenever possible.

In prior versions of SCS participation in SAMOS required a lot of user involvement. An event logger was required to be run for every cruise along with another 'mailer' application. However, in the current version of SCS, this workload has been reduced and an effort was made to make the SAMOS process as integrated, seamless and automatic as possible. That being said, a lot of user involvement is still required in terms of keeping meta-data up to date.

Meta Data Management

SAMOS metadata is managed inside the *CFE* page of SCS. Whenever a publish operation occurs all data fields of interest to SAMOS will automatically have calculated/derived feeds generated for them computing a 1 minute average. You do not have to create or manage any SAMOS definitions in this version of SCS. That being said, it is not recommended to edit the SAMOS sensor definitions either as when a publish operation occurs all existing SAMOS sensors are wiped clean and rebuilt from scratch (any changes you manually make to SAMOS interfaces, messages or datafields will be lost and reset on publish).

However, the meta data you enter with regards to the physical devices, vessel information and so forth DO change and are critical to the success of SAMOS evaluations. Please be sure to keep your physical inventory up to date and be sure to install / link all your logical interfaces to their appropriate physical devices so SAMOS is cognizant of what physical sensor is providing a given data stream in case manufacturing flaws or other useful information can be gleaned. While always important, for some devices this meta data is critical (calibration dates/files, etc) when it comes time to evaluate the produced streams for QA/QC.

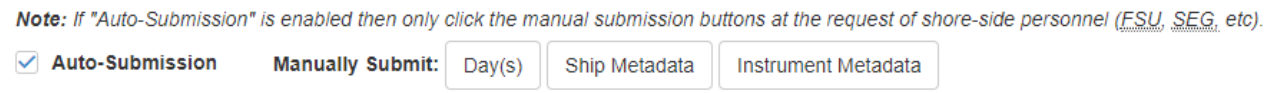
> ⚠ Do not manually modify any SAMOS interfaces, message definitions or data fields in CFE.

Submission Management

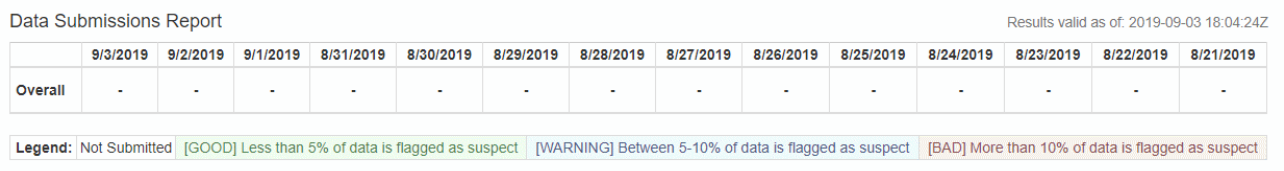
Submissions to SAMOS are reviewed and controlled via the *SAMOS* page under the *Data Management* menu item.

At the top of the page you will see a checkbox followed by a few buttons. The checkbox tells SCS to automatically extract, package and send your data to SAMOS every night. If this box is unchecked then no data will be sent to SAMOS unless you manually send it.

Manual submission of data is done via the buttons following the checkbox. You can submit sensor data (daily increments, hold control or shift to select multiple days or a range of days to send), current meta data regarding your vessel itself or current meta data covering your instrument suite as defined in CFE.

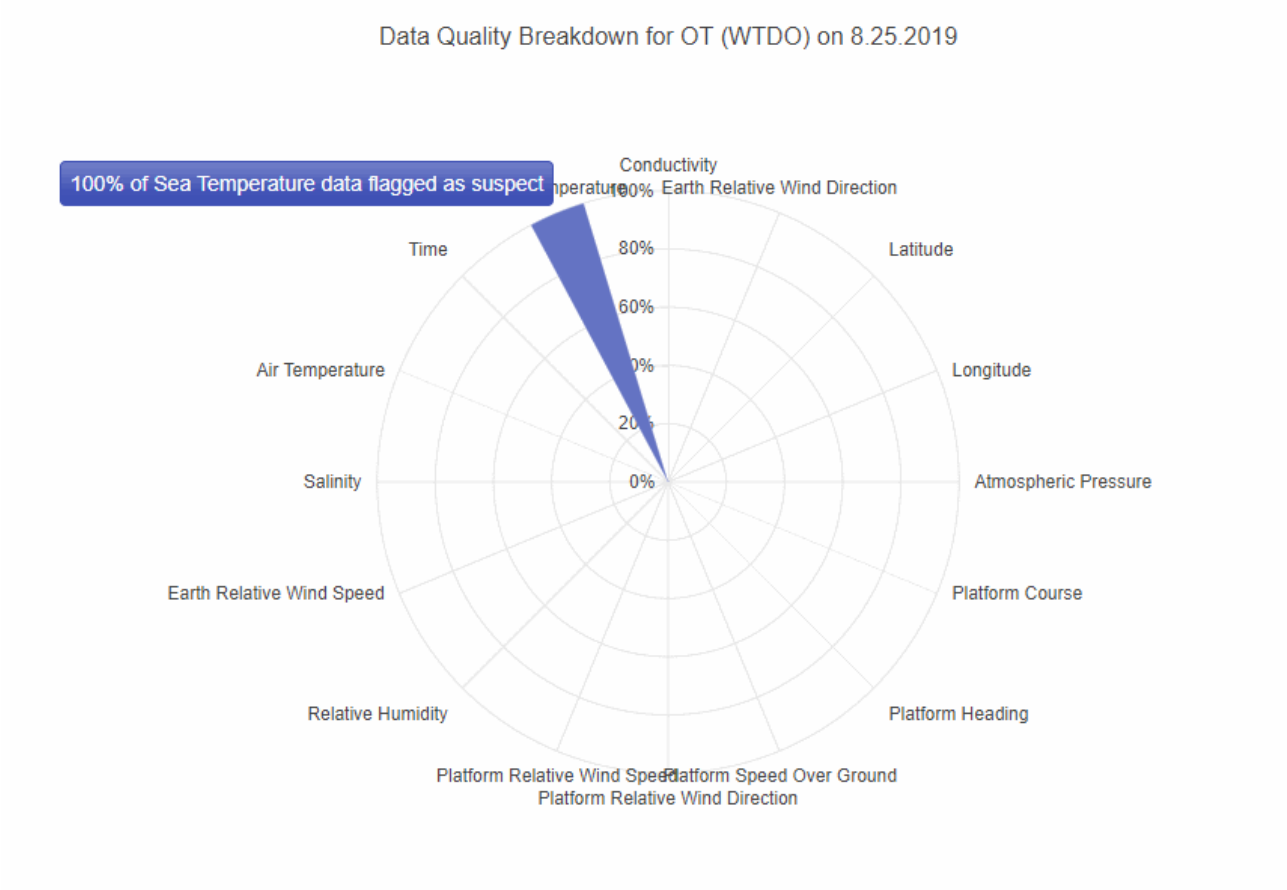


Below the submission buttons you will find a grid showing your submission history. You should review this to determine the quality of your data and to find gaps in your submissions. Each submitted day will be color coded to show you the overall QA/QC rating for the day.



If your ship is part of the NOAA fleet then more information can be found on the SDAT site - https://sdat.noaa.gov/DataManagement/DataQuality including what sensor is being flagged as problematic. This will enable you to resolve any issues SAMOS finds as early as possible rather then collecting bad data for the entire cruise before finding out there was a problem. Details regarding your daily submissions and their

QA/QC results can also be found on the SAMOS website itself. Your results will be available via the SAMOS site whether you are a NOAA ship or not.



NCEI

NOAA's National Centers for Environmental Information (NCEI) hosts and provides public access to one of the most significant archives for environmental data on Earth. Through the Center for Weather and Climate and the Center for Coasts, Oceans, and Geophysics, they provide over 25 petabytes of comprehensive atmospheric, coastal, oceanic, and geophysical data.

The interface located on via the *NCEI* page, found under the *Data Management* menu item, is intended for data submission to these national archives for SCS users on one of OMAO's large vessels. If you are not on one of these vessels please see your data coordinator for instructions on how to submit your SCS data (R2R, S2N, etc) to NCEI.

More details regarding your NOAA ship can be found on the SDALs Data Management page - https://sdat.noaa.gov/DataManagement/Tracking

Meta Data Management

The meta data provided to NCEI is pulled from the sensor configuration defined by CFE and the general system meta data you have described throughout SCS (such as your *vessel profile* page). You cannot directly send meta data to NCEI, however it is automatically pulled together and sent for each day that is submitted to NCEI, be it current or historic.

> ⚠ Meta-data defined outside CFE does not maintain a history, if you submit historic data the configuration of your sensor suite at that time will be sent, however your current ship metadata values (not those active at that time) will also be submitted. This may result in a mis-match should you wait to long to submit your datasets!

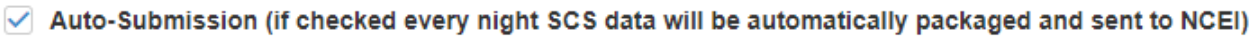
Submission Management

Submissions to NCEI can be reviewed and controlled via the ***NCEI*** page under the *Data Management* menu item.

The UI is broken down into two columns. On the left you will find your controls, a summary of your current status and the list of problematic prior submissions. On the right you will find a grid displaying all recent and active/current submission jobs.

Auto Submission

At the top of the control column you will find a checkbox which tells SCS to automatically submit the days data to NCEI every night. If this is checked then at midnight SCS will gather all sensor data collected (by SCS) for the prior 24 hours, package it up along with all relevant metadata and push it off to NCEI. If you know bandwidth will be limited, or you want to assume manual control, simply uncheck this box and SCS will no longer automatically submit data on your behalf. When you check the box again the auto submission will resume, however if you miss any submission windows you will have to manually send those days in as the software does not look back and fill in the gaps!

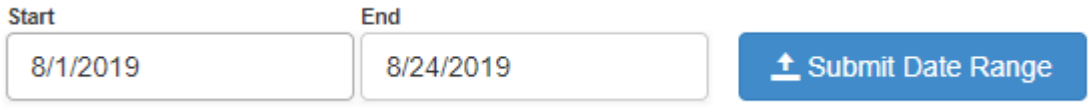


Manual Submission

To manually submit data all you have to do is enter a *Start* date and optionally an *End* date. Then click the *Submit Date Range* button.

> If you do not enter an end date then all days from the start date to 'now' will be [re]submitted.

> If you enter the same date (start date equals end date) then only that single day will be submitted.



Once you click the *Submit Date Range* button you should see the job appear in the submission grid located on the right.

Submission Summary

Below the auto and manual submission controls you will find a grid and graph depicting your ships current data submission status. Based upon your schedule in SDAT NCEI will expect a certain number of days worth of data from you. Obviously this number is only so accurate, deviations due to schedule changes and a multitude of other reasons will screw the number in one direction or the other. In those cases NCEI might reach out to you to [human to human] figure out what's going on and correct their expectations on their end.

Generally speaking however, in this section you can see a breakdown on how well your submitting your data. The prior year will be broken down into the categories defined below.

Missing
: Days the NCEI expected data which have not yet been sent in

Received
: Days sent in and acknowledged as having been received by NCEI, pending processing

Corrupted
: Days which made it into NCEI but were unable to be published due to errors

Published
: Days which have been officially processed and made available to the public.

In an ideal world all expected days will be *Published*, but in reality submissions might get corrupted in transit, days might get missed or deleted, etc. You can work with the OMAO data manager and/or NCEI themselves to resolve these issues as they occur.

Resubmission of Data

Below the submission summary you will find two lists. Inside these lists are the days which NCEI believe to be missing and the days which you submitted but NCEI found to be corrupted.

SCS will not automatically resubmit any of your data for you. Any days deemed to be missing (SCS was off, ships schedule changed, Auto-Submission was disabled, etc) will have to be resolved manually. Any corrupt days (zipping up the submission package didn't "zip" right, packets didn't transmit over the VSAT correctly, etc) will also have to be manually resolved. These flags are set on the shore, the only way to get them off the list is to have someone on shore clear them. This can be done by a variety of people, the primary POCs being the OMAO Data Manager or NCEI. Potentially Chief OPS at either MOC could also deal with it if schedule related.

The first step however is to simply submit or resubmit the days. This is easily accomplished by clicking the Submit Day button (or the Submit All button found at the bottom of the list). If a day is resubmitted and continues to present issues then at that point you would want to get someone on shore involved.

Once a day makes it to *Published* your responsibility for managing it has ended. That days data is now fully backed up and available on shore, you can and should delete them from your ship as they can be retrieved from the National Archives as needed.

Recent and Active Submissions

On the right hand side of the page you will find a grid listing all recent and currently in-progress submission jobs for NCEI. Each job is a days worth of data and can be in one of 4 states.

Date	Extracting	Packaging	Transferring	
September 01, 2019	In Progress	Waiting	Waiting	Abort
August 31, 2019	In Progress	Waiting	Waiting	Abort
August 30, 2019	In Progress	Waiting	Waiting	Abort
August 29, 2019	In Progress	Waiting	Waiting	Abort

A spinning cog icon will tell you which state each job (a row / date) is in. When complete it will change to a green checkbox and move to the next step.

The first step is extraction of the datasets from the database. This is essentially a data dump for the entire day and could take some time depending on how large a sensor suite you have and how much data you recorded for that day.

After the data is extracted it is packaged (generally into a NetCDF format along with various XML encoded files) in a manner NCEI understands and can ingest, it is then zipped up in preparation for transmission.

When the package is built it can be sent to NCEI a variety of ways. This could be very fast if simply transferred locally for a ship-to-shore sync (Aspera, rSync, NetMan, etc) or sent over a high speed connection. It could also take quite some time if actually sent to shore over the VSAT or other low speed connection.

⚠ As of the time of this manual being written there is no way for you, as a general user, to decide which mechanism is used to send the data to NCEI. It is still pending discussions between OMAO, MO, the other line offices and NCEI.

Once all columns are marked in green then the job is completed and you can look forward to the day no longer being marked as *Missing* by NCEI.

If a job gets screwy or you want to cancel it for whatever reason you can always abort it via the button in the last column.

Thermosalinograph

NOAA supports the collection of sea surface salinity (SSS) and sea surface temperature (SST) data from thermosalinographs installed on ships of the NOAA fleet and ships of the Ship Of Opportunity Program (SOOP). Both SSS and SST observations from TSGs are part of the Global Ocean Observing System.

The global atmospheric and oceanic observations, including TSG observations from ship under the SOOP and research vessels, have been the foundation for understanding long-term changes in marine climate and are essential input to climate and weather forecast models.

It is requested that, if possible, ships submit their thermosalinograph (TSG) data to NOAA's Atlantic Oceanographic and Meteorological Laboratory (AOML). SCS has an automated system built in to accomplish this as described below.

More information can be found here - https://www.aoml.noaa.gov/phod/tsg/background.php

Meta Data Management

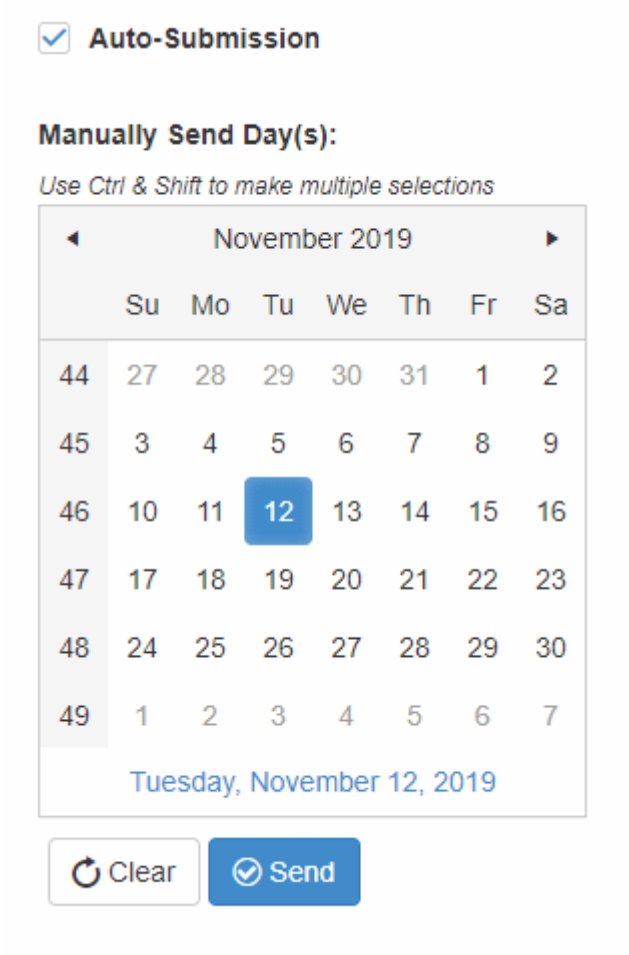
TSG metadata is managed inside the *CFE* page of SCS. Please review the physical inventory portion of *CFE* and ensure all related metadata (including any calibration files) are up to date inside SCS. Much of the metadata sent to AOML will come from the values you have set in *CFE*. Additional, though perhaps less dynamic, data is obtained from the *Vessel Profile* portion of SCS. It is a good idea to periodically review those meta data items to ensure they remain accurate as well.

Submission Management

Submissions to AOML are reviewed and controlled via the ***TSG*** page under the *Data Management* menu item.

At the top of the page you will see a checkbox, this tells SCS to automatically extract, package and send your TSG data to AOML every night. If this box is unchecked then no data will be sent to AOML unless you manually send it.

Manual submission of data is done via the calendar and buttons below the checkbox. You can submit sensor data (daily increments, hold control or shift to select multiple days or a range of days to send) by selecting the days to send and clicking the *Send* button.



Data Extraction

Sensor data in SCS is collected by the primary acquisition service ACQ. ACQ then writes the data out to a number of different targets for ingestion/use by other external parties or subsystems inside SCS itself. These targets are subject to change as SCS evolves, but at the time of this writing they are as follows:

- SignalR
- Database
- Files
- UDP
- NetCDF

> ⚠ Note: NetCDF has been temporarily removed from SCS pending formatting agreements between OMAO and NCEI. Though present in this documentation it will not be available for use in SCS until the agreements are in place.

SignalR

SignalR is an open-source library that simplifies adding real-time web functionality to apps. Real-time web functionality enables server-side code to push content to clients instantly. It provides an API for creating

server-to-client remote procedure calls (RPC). The RPCs call JavaScript functions on clients from server-side .NET code. More information can be found in the SignalR section of the help.

Database

The backend of SCS is a MySQL database and is the backbone of the entire system. In regards to this section the database is the home of all recorded data ranging from sensor data collected by ACQ and event data collected by the Event Logger to various system logs and beyond. The database is meant to be non-interactive from a user perspective and is simply an internal store for SCS itself. For instance, to get data from the database SCS provides various tools for your use rather then requiring you to run SQL queries yourself. Sensor data stored in the database can be extracted via the *Extraction / To Disk* page located in the *Data Management* menu. Instructions to do this are below in the NetCDF section.

Files

In the past SCS has written data directly to disk in the form of RAW files. The current version of SCS does this as well for backwards compatibility as many other 3rd party apps have been written around these files and time will be needed for them to be updated. Long term the target input for these apps should migrate over to NetCDF however as RAW files will be rendered obsolete at some point in the future.

RAW files are direct dumps of the sensor feeds to a file. These files are located in optional sub-folders as defined by the user via CFE (default root is D:\SCS\DATA\RAW DATA). Each file corresponds to a single days worth of data from a single message. As it's a direct write of the sensor feed it's impossible to tell what the format of the data is, however in the majority of cases it is NMEA compliant and could be considered a comma-delimited data-set.

> ⚠ The latest SCS adds a header line to each RAW file that was not present in prior versions of SCS. Please modify any ingestion routines to accommodate!

In the event you wish to modify the timestamp used when generating the file names you can do so by manually modifying the .config file for the ACQ service. Inside the file there is a setting named OutputFileNameFormat whose default value is '3'.

```
<add key="OutputFileNameFormat" value="3" />
```

By modifying the value (and restarting ACQ) you will set the naming convention used for the RAW files as detailed below. Note this does not effect the timestamp of the RAW data itself which is locked into the ISO 8601 format.

Value	Example Result
0	2019-02-08 19.25.24Z SIMRAD MX512 GPS-MX512 #2 GPS-$GPGGA.RAW.log
1	SIMRAD MX512 GPS-MX512 #2 GPS-$GPGGA 2019-02-08 19.25.24Z.RAW.log
2	1550843658_SIMRAD_MX512_GPS-MX512_#2_GPS-$GPGGA.RAW.log
3	20190208T192524Z_SIMRAD_MX512_GPS-MX512_#2_GPS-$GPGGA.RAW.log

UDP

ACQ automatically sends data out via UDP Broadcast if an admin has requested it to do so. Details on this are described in the UDP section.

NetCDF

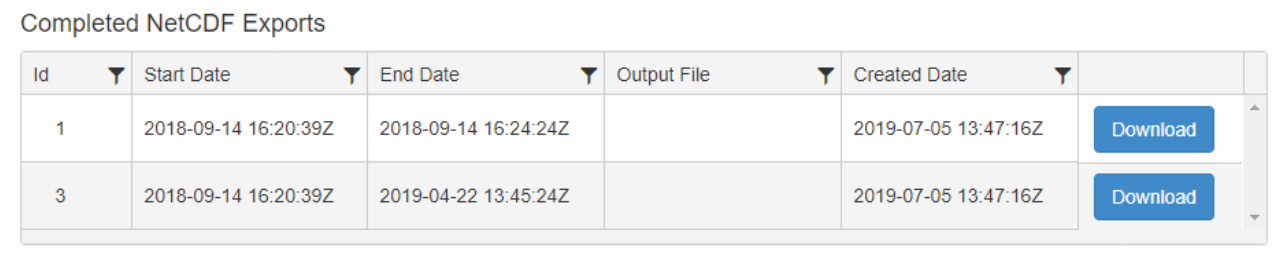
> ⚠ See note above regarding temporary removal of NetCDF functionality from SCS.

NetCDF is the format most data pulled from SCS should be in. It has become the format requested by NCEI and many scientists and so SCS has switched from RAW files to NetCDF as the primary output. SCS writes sensor data to NetCDF both in real time and as requested by users. SCS complies with the templates set forth by NCEI (https://www.nodc.noaa.gov/data/formats/netcdf/v2.0/) however as we do not wish to interpolate data we cannot use any feature type other than a point. Unfortunately this may not be considered very useful by end users. Future versions of SCS *may* add an interpolated export and/or NCEI themselves may run routines to take the RAW SCS NetCDF and generate a more useful output.

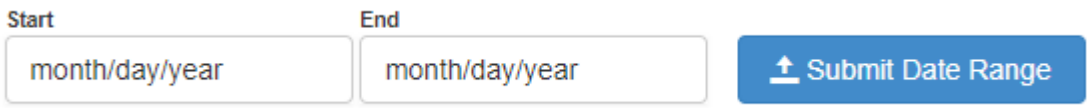
There are many tools out there for opening and viewing NetCDF files. Please work with your system administrator to see which one you should use. If you are an OMAO user the one on the approved software list is Panoply - https://www.giss.nasa.gov/tools/panoply/

As ACQ writes data to the above outputs SCS also starts building out a NetCDF version of the data in real time. While this is redundant it does save time post-cruise when an entire cruise worth of data is needed since the dumps will have already been created. However, in the event the real-time files fail or have errors SCS also allows manual building and extraction requests to be issued to pull the data directly from the database and dump it to disk in NetCDF format.

To download daily sensor data NetCDF files open the *Extraction / To Disk* page located in the *Data Management* menu. A list of completed NetCDF files will be displayed. If you want to pull one simply click the *Download* button at the end of it's row.

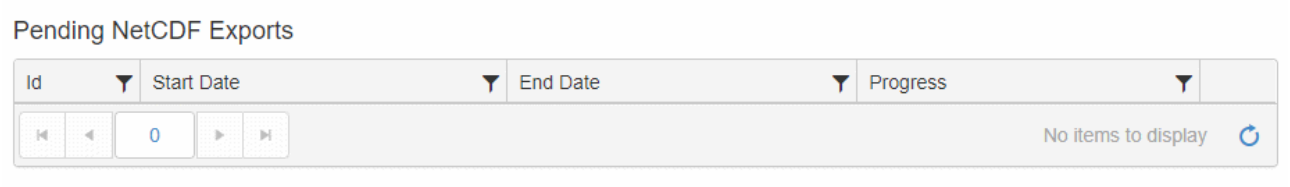


If the date you want is not present in the list (and is not pending on the list on the right) you can submit a new job to extract and build NetCDF files by entering a start/end date range and clicking the *Submit* button.



If you are a server admin and have access to the filesystem the NetCDF files are stored by default in the D:\SCS\DATA\NetCdf folder.

NetCDF files that are in the process of being built will be displayed in the 'Pending' grid on the right side of the screen. Once they've completed the files will become available for download.



> Much like hitting a button in an elevator, repeatedly hammering it / adding the same day over and over will not actually speed up the processing job so please be patient.

Data Cleanup

To manually delete stale data from your system you can do so via the *Data Cleanup* menu item under *Data Management*.

Click the calendar icon to choose a date and the clock icon to specify a time. Then click the *Clear Old Data* button to start the purge.

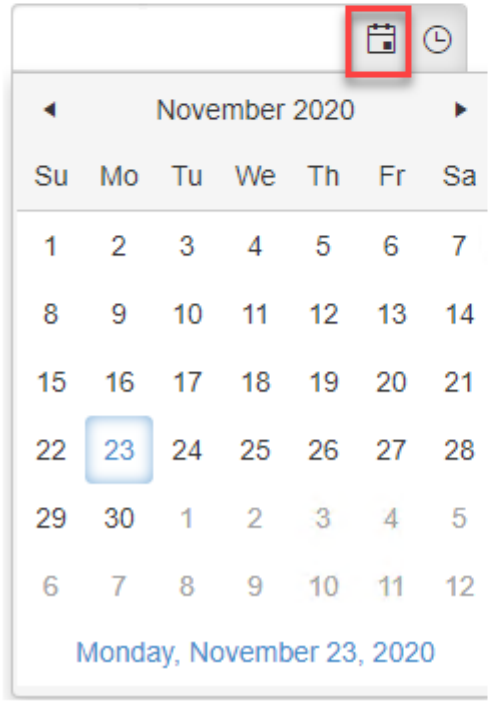
Data Cleanup

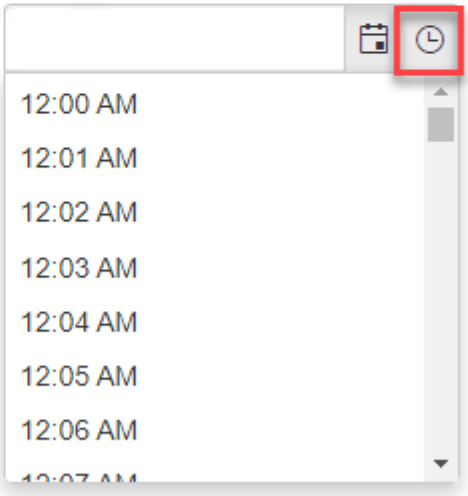
Manually Clear

Clear old data prior to selected date and time from the selected targets

Target	Clear Old Data?
Database Sensor Data	☑
Raw Data Files	☑
Data Exports	☑
Event Data	☑

Clear Old Data





⚠ This is a permanent operation, deleting data in this manner is non-reversible.

SCS automatically cleans old data for you so this operation shouldn't be nessecary though it might be a good idea to run it annually just to be sure.

⚠ Depending on the amount of data there is in your system this could be a long running operation. It is advisable to kick the job off when you are on shore or SCS is not actively being used (potential performance impacts).

SignalR

SignalR is an open-source library that simplifies adding real-time web functionality to apps. Real-time web functionality enables server-side code to push content to clients instantly. It provides an API for creating server-to-client remote procedure calls (RPC). The RPCs call JavaScript functions on clients from server-side .NET code. SCS uses SignalR as the backplane to supply clients with real-time data and basic communications. Various widgets such as Charts and Real Time Displays get their data via the SignalR "Data Hub". Systems are notified of template changes, CFE configuration changes and other such events of note by subscribing to the "Configuration Hub", and so forth. SignalR plays a critical role in the real-time aspect of SCS.

> Critical communications (such as interactions with the database or important commands issues to the other subsystems like Events) do not rely upon SignalR and instead use robust protocols/frameworks such as TCP and WCF.

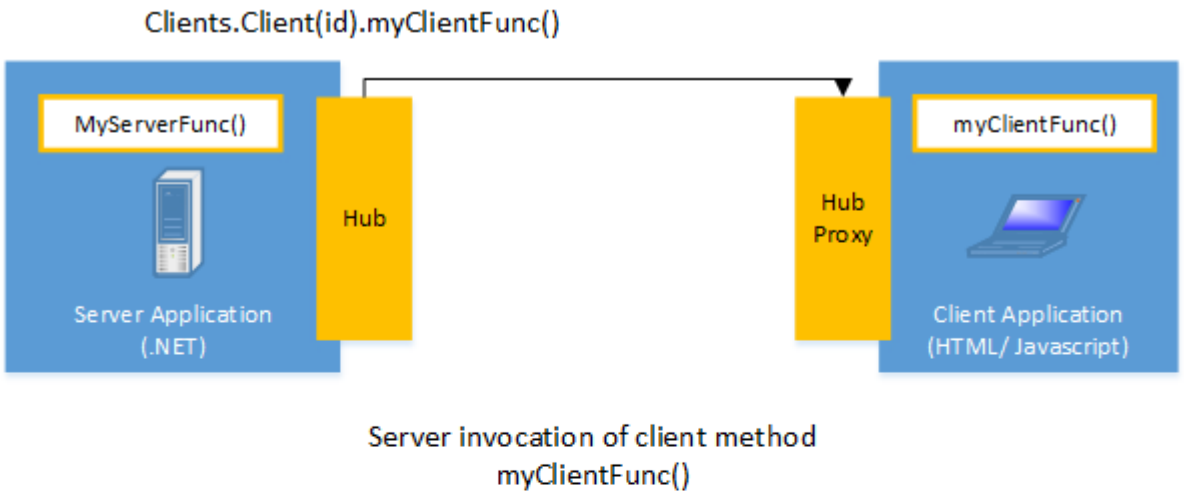
What is SignalR?

ASP.NET SignalR is a library for ASP.NET developers that simplifies the process of adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.

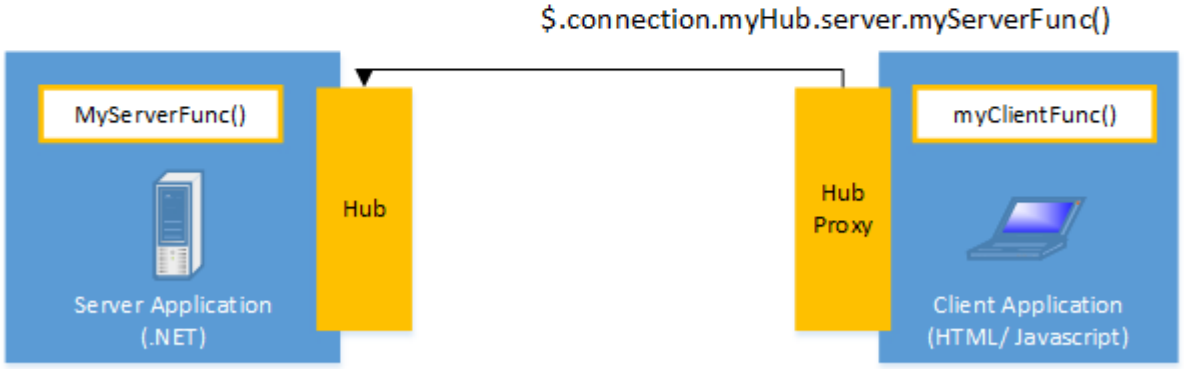
SignalR can be used to add any sort of "real-time" web functionality to your ASP.NET application. While chat is often used as an example, you can do a whole lot more. Any time a user refreshes a web page to see new data, or the page implements long polling to retrieve new data, it is a candidate for using SignalR. Examples include dashboards and monitoring applications, collaborative applications (such as simultaneous editing of documents), job progress updates, and real-time forms.

SignalR also enables completely new types of web applications that require high frequency updates from the server, for example, real-time gaming.

SignalR provides a simple API for creating server-to-client remote procedure calls (RPC) that call JavaScript functions in client browsers (and other client platforms) from server-side .NET code. SignalR also includes API for connection management (for instance, connect and disconnect events), and grouping connections.



Server invocation of client method
myClientFunc()



Client invocation of server method
MyServerFunc()

SignalR handles connection management automatically, and lets you broadcast messages to all connected clients simultaneously, like a chat room. You can also send messages to specific clients. The connection between the client and server is persistent, unlike a classic HTTP connection, which is re-established for each communication.

SignalR supports "server push" functionality, in which server code can call out to client code in the browser using Remote Procedure Calls (RPC), rather than the request-response model common on the web today.

SignalR applications can scale out to thousands of clients using Service Bus, SQL Server or Redis.

SignalR is open-source, accessible through GitHub.

SignalR and WebSocket

SignalR uses the new WebSocket transport where available and falls back to older transports where necessary. While you could certainly write your app using WebSocket directly, using SignalR means that a lot of the extra functionality you would need to implement is already done for you. Most importantly, this means that you can code your app to take advantage of WebSocket without having to worry about creating a separate code path for older clients. SignalR also shields you from having to worry about updates to WebSocket, since SignalR is updated to support changes in the underlying transport, providing your application a consistent interface across versions of WebSocket.

Transports and fallbacks

SignalR is an abstraction over some of the transports that are required to do real-time work between client and server. A SignalR connection starts as HTTP, and is then promoted to a WebSocket connection if it is available. WebSocket is the ideal transport for SignalR, since it makes the most efficient use of server memory, has the lowest latency, and has the most underlying features (such as full duplex communication between client and server), but it also has the most stringent requirements: WebSocket requires the server to be using Windows Server 2012 or Windows 8, and .NET Framework 4.5. If these requirements are not met, SignalR will attempt to use other transports to make its connections.

HTML 5 transports

These transports depend on support for HTML 5. If the client browser does not support the HTML 5 standard, older transports will be used.

- WebSocket (if the both the server and browser indicate they can support Websocket). WebSocket is the only transport that establishes a true persistent, two-way connection between client and server. However, WebSocket also has the most stringent requirements; it is fully supported only in the latest versions of Microsoft Internet Explorer, Google Chrome, and Mozilla Firefox, and only has a partial implementation in other browsers such as Opera and Safari.
- Server Sent Events, also known as EventSource (if the browser supports Server Sent Events, which is basically all browsers except Internet Explorer.)

Comet transports

The following transports are based on the Comet web application model, in which a browser or other client maintains a long-held HTTP request, which the server can use to push data to the client without the client specifically requesting it.

- Forever Frame (for Internet Explorer only). Forever Frame creates a hidden IFrame which makes a request to an endpoint on the server that does not complete. The server then continually sends script to the client which is immediately executed, providing a one-way realtime connection from server to client. The connection from client to server uses a separate connection from the server to client connection, and like a standard HTTP request, a new connection is created for each piece of data that needs to be sent.
- Ajax long polling. Long polling does not create a persistent connection, but instead polls the server with a request that stays open until the server responds, at which point the connection closes, and a new connection is requested immediately. This may introduce some latency while the connection resets.

For more information on what transports are supported under which configurations, see Supported Platforms.

Transport selection process

The following list shows the steps that SignalR uses to decide which transport to use.

1. If the browser is Internet Explorer 8 or earlier, Long Polling is used.
2. If JSONP is configured (that is, the `jsonp` parameter is set to `true` when the connection is started), Long Polling is used.
3. If a cross-domain connection is being made (that is, if the SignalR endpoint is not in the same domain as the hosting page), then WebSocket will be used if the following criteria are met:

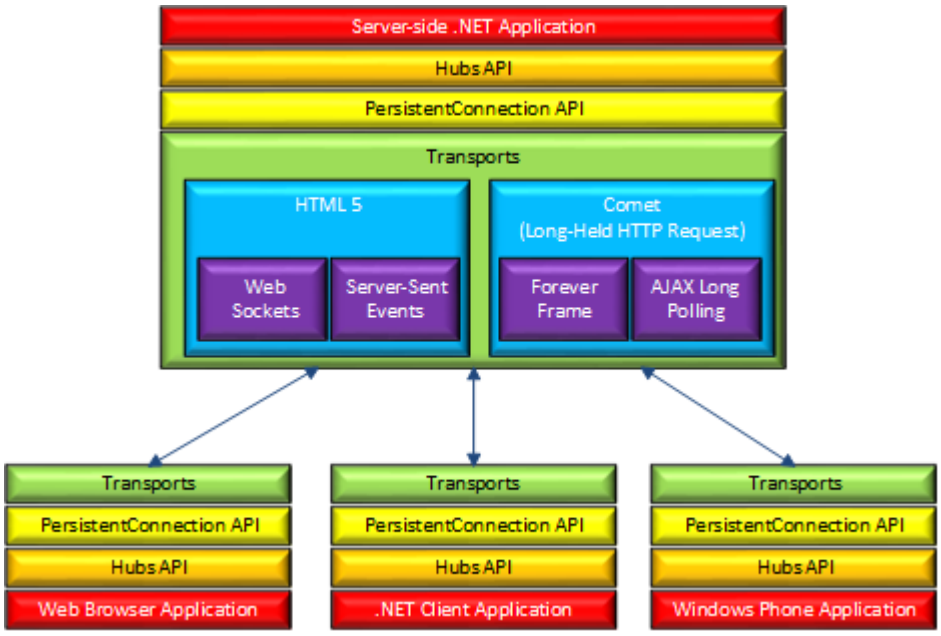
- The client supports CORS (Cross-Origin Resource Sharing). For details on which clients support CORS, see CORS at caniuse.com.
- The client supports WebSocket
- The server supports WebSocket

  If any of these criteria are not met, Long Polling will be used. For more information on cross-domain connections, see How to establish a cross-domain connection.

4. If JSONP is not configured and the connection is not cross-domain, WebSocket will be used if both the client and server support it.
5. If either the client or server do not support WebSocket, Server Sent Events is used if it is available.
6. If Server Sent Events is not available, Forever Frame is attempted.
7. If Forever Frame fails, Long Polling is used.

Architecture diagram

The following diagram shows the relationship between Hubs, Persistent Connections, and the underlying technologies used for transports.



How Hubs work

When server-side code calls a method on the client, a packet is sent across the active transport that contains the name and parameters of the method to be called (when an object is sent as a method parameter, it is serialized using JSON). The client then matches the method name to methods defined in client-side code. If there is a match, the client method will be executed using the deserialized parameter data.

The method call can be monitored using tools like Fiddler. The following image shows a method call sent from a SignalR server to a web browser client in the Logs pane of Fiddler. The method call is being sent from a hub called `MoveShapeHub`, and the method being invoked is called `updateShape`.

```
10:38:03:1298 Session846.WebSocket'WebSocket #846' - Pushing 104 bytes from server WebSocket
81 66 7B 22 43 22 3A 22 42 2C 31 35 7C 43 2C 30     f{"C":"B,15|C,0
7C 44 2C 30 7C 45 2C 30 22 2C 22 4D 22 3A 5B 7B  |D,0|E,0","M":[{
22 48 22 3A 22 4D 6F 76 65 53 68 61 70 65 48 75  "H":"MoveShapeHu
62 22 2C 22 4D 22 3A 22 75 70 64 61 74 65 53 68  b","M":"updateSh
61 70 65 22 2C 22 41 22 3A 5B 7B 22 6C 65 66 74  ape","A":[{"left
22 3A 35 30 31 2E 30 2C 22 74 6F 70 22 3A 33 30  ":501.0,"top":30
32 2E 30 7D 5D 7D 5D 7D                          2.0}]}]}
```

In this example, the hub name is identified with the `H` parameter; the method name is identified with the `M` parameter, and the data being sent to the method is identified with the `A` parameter. The application that generated this message is created in the High-Frequency Realtime tutorial.

Choosing a communication model

Most applications should use the Hubs API. The Connections API could be used in the following circumstances:

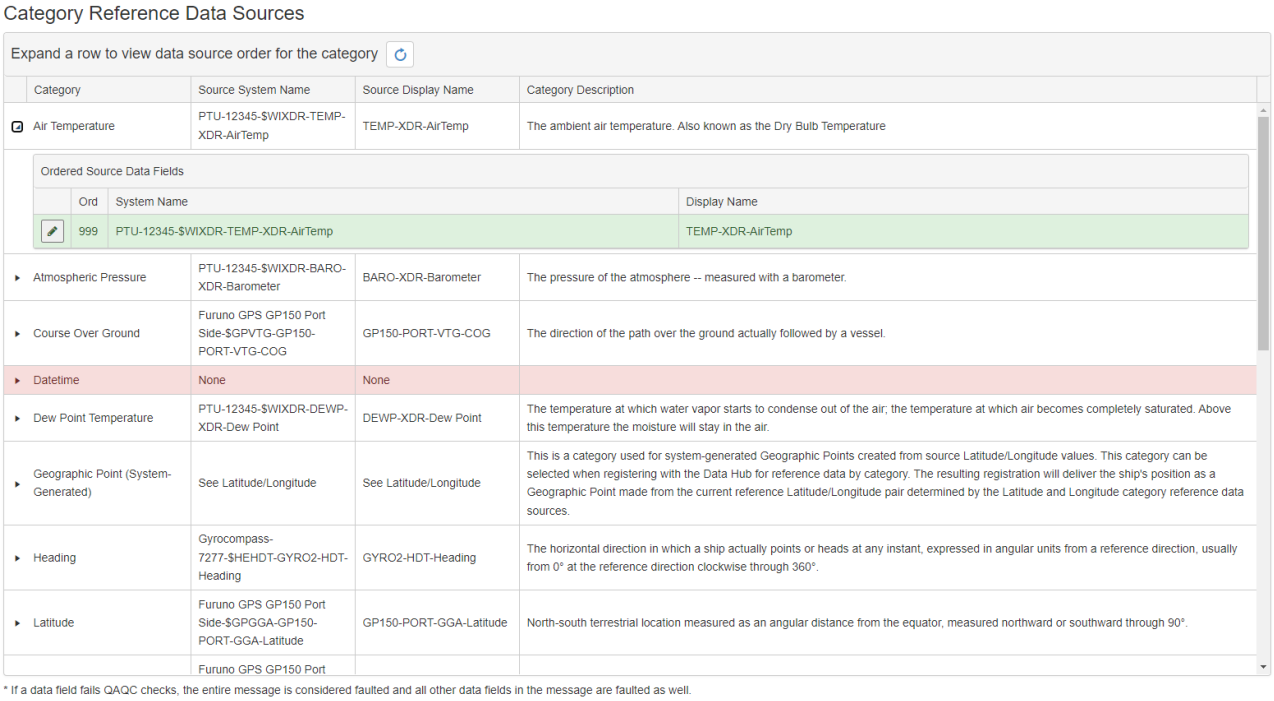
- The format of the actual message sent needs to be specified.
- The developer prefers to work with a messaging and dispatching model rather than a remote invocation model.
- An existing application that uses a messaging model is being ported to use SignalR.

More information and much of the above can be found here - https://docs.microsoft.com/en-us/aspnet/signalr/

Reference Sources

As detailed in the CFE portion of the manual, some Data Field Categories can be referenced as a source of data rather than directly selecting a sensor's data field. For instance you might have 3 GPS units with varying levels of accuracy or preference. By referencing the Latitude category SCS will automatically provide best GPS as a source at any given moment.

For obvious reasons there are plenty of use cases where the above is preferable. However, when referencing a category you aren't always aware which sensor is truly providing you the data. To get a holistic view you can reference this portion of the *Settings* page to determine WHICH sensor is providing the feed for each data field category at the moment.


Category Reference Data Sources

Expand a row to view data source order for the category

Category	Source System Name	Source Display Name	Category Description
Air Temperature	PTU-12345-$WIXDR-TEMP-XDR-AirTemp	TEMP-XDR-AirTemp	The ambient air temperature. Also known as the Dry Bulb Temperature
Atmospheric Pressure	PTU-12345-$WIXDR-BARO-XDR-Barometer	BARO-XDR-Barometer	The pressure of the atmosphere -- measured with a barometer.
Course Over Ground	Furuno GPS GP150 Port Side-$GPVTG-GP150-PORT-VTG-COG	GP150-PORT-VTG-COG	The direction of the path over the ground actually followed by a vessel.
Datetime	None	None	
Dew Point Temperature	PTU-12345-$WIXDR-DEWP-XDR-Dew Point	DEWP-XDR-Dew Point	The temperature at which water vapor starts to condense out of the air; the temperature at which air becomes completely saturated. Above this temperature the moisture will stay in the air.
Geographic Point (System-Generated)	See Latitude/Longitude	See Latitude/Longitude	This is a category used for system-generated Geographic Points created from source Latitude/Longitude values. This category can be selected when registering with the Data Hub for reference data by category. The resulting registration will deliver the ship's position as a Geographic Point made from the current reference Latitude/Longitude pair determined by the Latitude and Longitude category reference data sources.
Heading	Gyrocompass-7277-$HEHDT-GYRO2-HDT-Heading	GYRO2-HDT-Heading	The horizontal direction in which a ship actually points or heads at any instant, expressed in angular units from a reference direction, usually from 0° at the reference direction clockwise through 360°.
Latitude	Furuno GPS GP150 Port Side-$GPGGA-GP150-PORT-GGA-Latitude	GP150-PORT-GGA-Latitude	North-south terrestrial location measured as an angular distance from the equator, measured northward or southward through 90°.
	Furuno GPS GP150 Port		

Ordered Source Data Fields

Ord	System Name	Display Name
999	PTU-12345-$WIXDR-TEMP-XDR-AirTemp	TEMP-XDR-AirTemp

* If a data field fails QAQC checks, the entire message is considered faulted and all other data fields in the message are faulted as well.

The first column of the grid will have the Data Field Category being referenced. The second and third columns will tell you which actual Data Field SCS is currently using to serve up the sensor data stream for the given category.

You can expand each row/category to see the list of potential sources, the one highlighted in green is the currently active source.

⚠ If - *None* - is displayed then there are no Data Fields found in your configuration which have their category set to the given type.

⚠ If - *Note Available* - is displayed then there are Data Fields found in your configuration which have their category set to the given type but none of them are currently good enough (eg failing QA/QC checks) to act as a reference source.

⚠ If the category name is red then that category has no valid feed, do not use it as a reference in your templates until you assign at least one Data Field to it.

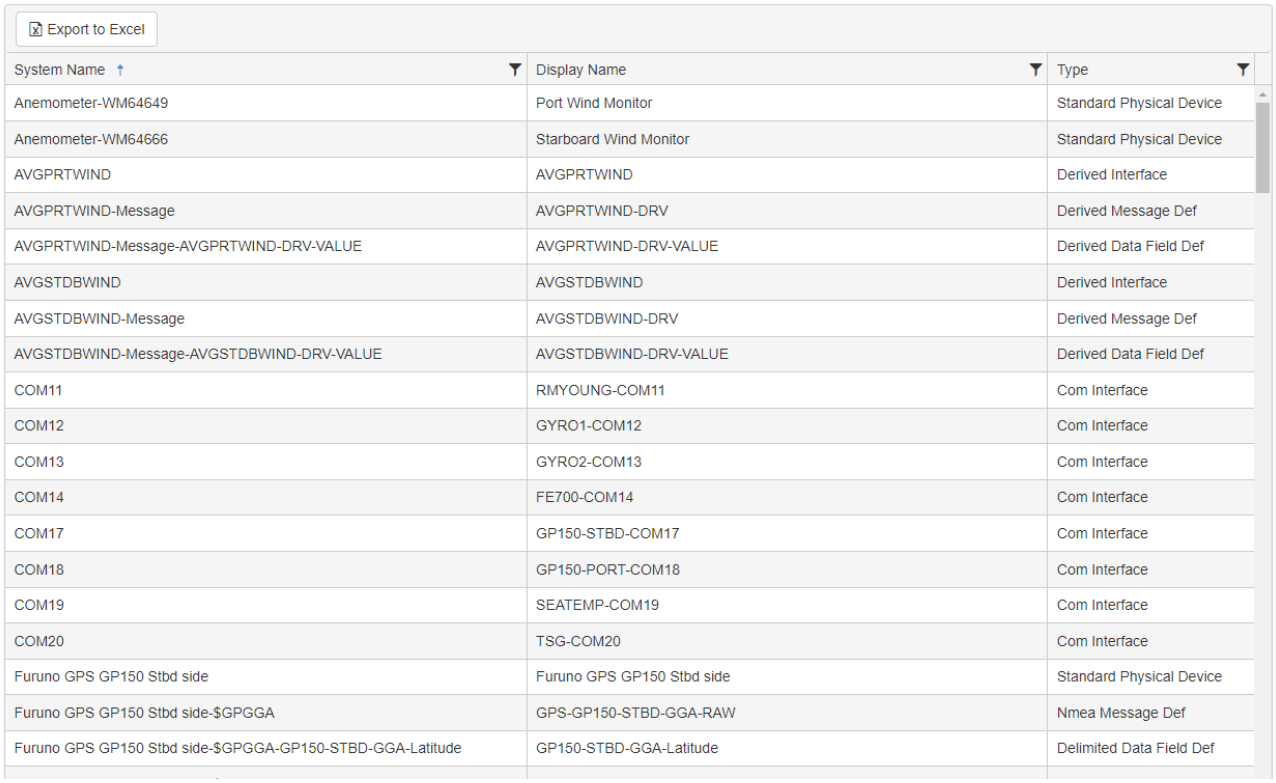
To modify which data fields are eligible for a Data Field Category reference you must use CFE.

See also: CFE - Physical Device Reference Order

Name Mapping

When a new sensor is created inside CFE it is automatically assigned a system generated name. This helps with cross-referencing and data discovery once the data makes it to the archives and is [potentially] combined with data from other ships. The system name is not very user friendly however which is why SCS provides the ability for the ship to create *Display Name*s as an alternate name to be used in various widgets, visualizations and systems inside SCS.

The *Name Mapping* page provides users a way to quickly see a list of all sensor configuration items and how their system generated name maps to the user defined display name. This makes it easier to determine what exactly you are looking at if you're only given one or the other naming convention. The data can be exported to excel via the button at the top of the grid for use outside of SCS.


Export to Excel

System Name	Display Name	Type
Anemometer-WM64649	Port Wind Monitor	Standard Physical Device
Anemometer-WM64666	Starboard Wind Monitor	Standard Physical Device
AVGPRTWIND	AVGPRTWIND	Derived Interface
AVGPRTWIND-Message	AVGPRTWIND-DRV	Derived Message Def
AVGPRTWIND-Message-AVGPRTWIND-DRV-VALUE	AVGPRTWIND-DRV-VALUE	Derived Data Field Def
AVGSTDBWIND	AVGSTDBWIND	Derived Interface
AVGSTDBWIND-Message	AVGSTDBWIND-DRV	Derived Message Def
AVGSTDBWIND-Message-AVGSTDBWIND-DRV-VALUE	AVGSTDBWIND-DRV-VALUE	Derived Data Field Def
COM11	RMYOUNG-COM11	Com Interface
COM12	GYRO1-COM12	Com Interface
COM13	GYRO2-COM13	Com Interface
COM14	FE700-COM14	Com Interface
COM17	GP150-STBD-COM17	Com Interface
COM18	GP150-PORT-COM18	Com Interface
COM19	SEATEMP-COM19	Com Interface
COM20	TSG-COM20	Com Interface
Furuno GPS GP150 Stbd side	Furuno GPS GP150 Stbd side	Standard Physical Device
Furuno GPS GP150 Stbd side-$GPGGA	GPS-GP150-STBD-GGA-RAW	Nmea Message Def
Furuno GPS GP150 Stbd side-$GPGGA-GP150-STBD-GGA-Latitude	GP150-STBD-GGA-Latitude	Delimited Data Field Def

Settings

System Settings can be accessed in the ***Settings*** page located in the *System* main menu item. The majority of these items are read-only and are changed elsewhere in the system, however the various settings are consolidated here so you can get a good overview of all.

> The main exceptions to this are the email settings and the auto-submission time settings, which are viewed and edited via this page.

General Settings

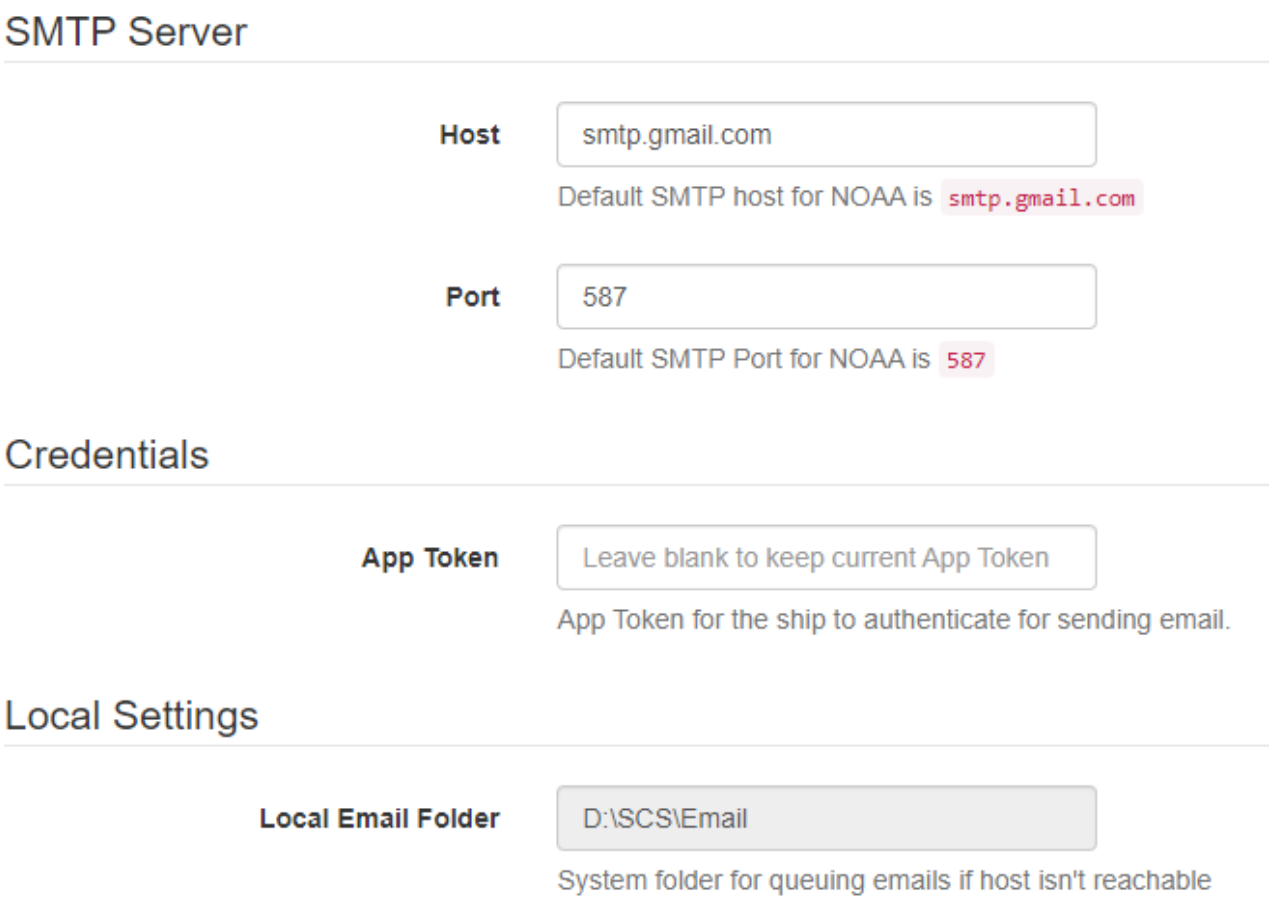
The general settings portion of the page takes various settings from all areas of SCS and displays them in a grid. The first column is the source area for the given setting, if you wish to change the setting then this would be a clue as to where you should go to access it. The next column is the setting name itself followed by the last column which is the settings current value.

Area	Setting	Value
Application	CurrentShipID	11
Application	OverrridenCurrentShipID	False
Database	CurrentShipID	11
Database	DefaultSCSFolder	D:\SCS
Disseminator	Disseminator.HubBaseUrl	http://localhost
Disseminator	SDAT.BaseURL	omao-seg-web1
Disseminator	Disseminator.EmailRetryInterval_minutes	15
Disseminator	SCS.EmailFolder	D:\SCS\Email
Disseminator	NCEI.Enabled	false
Disseminator	SAMOS.Enabled	false

Obviously not all configuration items and settings are displayed here, only the most relevant ones. Some of these are not meant to be modified and are simply so you are aware of their value (the SCS version number for example).

Email

The one setting you are most likely to change here is email. At the bottom of the page you will see a link which will direct you to a page allowing you to update the email settings for the entire SCS system.



The first email setting you are presented with allows you to specify which outbound SMTP mail host you wish to use. If you have a ship or shore based server enter it here, if you use a cloud provider such as GMAIL consult them for what values to use.

The next value is the port to use - 587, 465 and 25 are common values to find here.

Unless you have an anonymous email server set as your host you will most likely need to supply credentials to send email. In this section you can provide those credentials.

If you are a NOAA ship you should be issued an email app token by SEG.

If you are NOT a large NOAA ship you can enter a username and password to use when connecting to the SMTP host and sending email.

Note these are your email credentials, not your computer/AD credentials. If you are using 2 factor authentication for your email server you may need to create application specific passwords for SCS. Consult with your vendor for more information.

SCS will attempt to send emails immediately, however should that fail for whatever reason (bandwidth, etc) the email will be saved to disk and continually tried again until it goes through. This is the folder those emails will be saved to pending send.

> Emails serialized to disk are encrypted, even if you are able to access this folder you cannot view or change the emails awaiting to be sent.

> Do not forget to apply your changes for them to take effect!

See also: App Passwords in Gmail (general information)

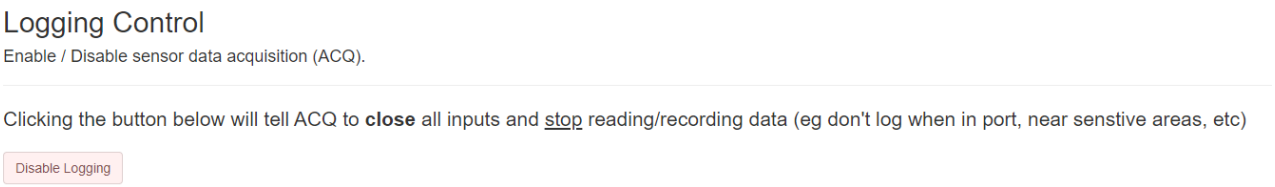
See also: Creating App Passwords in Gmail

Submission Settings

If you have the daily submission modules enabled you can control when they fire off their packages. It is a good idea to keep these somewhat temporally spread so they are not all clogging up the system resources at the same time.

Logging Control

The ability to start and stop ACQ from logging data can be manually controlled by clicking the *Logging Control* link in the *System* menu.



If ACQ logging is disabled then a warning icon will flash on the main menu until it is re-enabled.

> Note logging can also be started and stopped via GIS Triggers

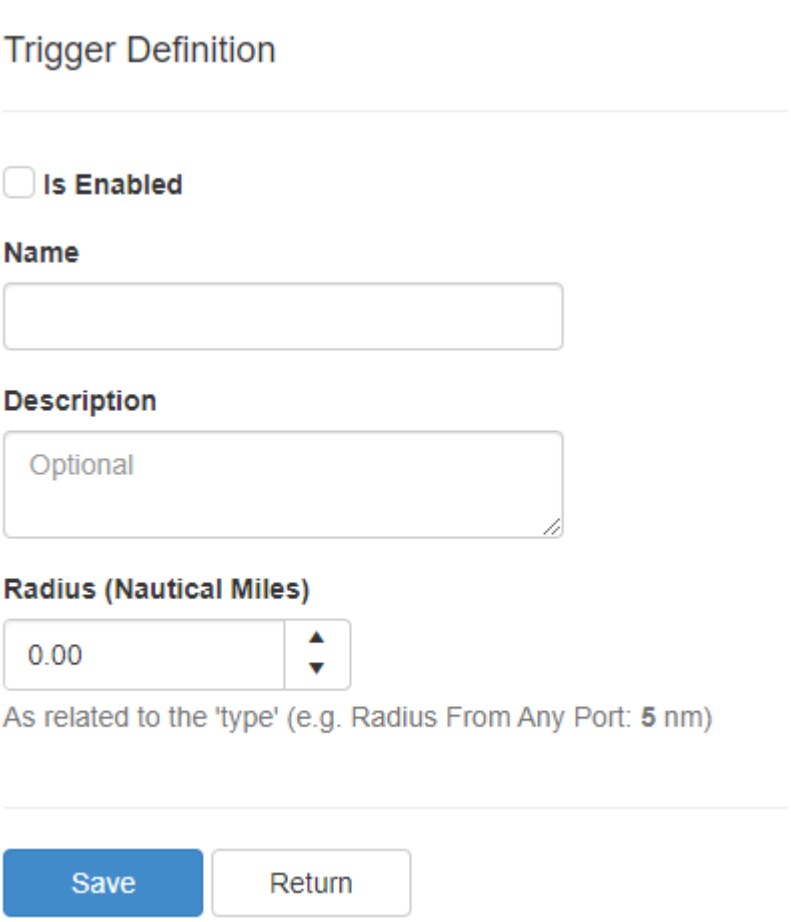
GIS Triggers

SCS offers a way to trigger actions inside the system based upon your ships current location in the world and the direction it is traveling. For instance, you can setup a set of actions to occur whenever you leave your home port and a different set to occur when you come back home. A set of known ports comes with the SCS system, however you can also trigger off a custom geographic point (latitude / longitude coordinate) if your port is not listed or your trigger isn't meant to be based off a port (eg say you want to trigger whenever you are near a certain sea mount or some other feature).

You can setup and manage your spatial triggers via the *GIS Triggers* page located under the *System* menu item. There are 3 main types of triggering mechanisms; you can trigger when you are around any port, a specific port or a custom point (latitude/longitude). To create a trigger simply click the *+Add* button located on the top of the corresponding grid.

Each type shares a common set of parameters and a very similar builder. The builder UI consists of 2 columns, on the left you define the trigger itself and on the right you define the actions the trigger should kick off when it fires.

Trigger Definition



When checked SCS will enable this trigger and actively monitor the vessels position to see how it relates to the values you have specified here.

Each trigger must have a *Name*, this can be anything you like though it should describe what you are generally looking to accomplish.

Should you wish to provide further details to remind yourself or others what the point of this trigger is and why it exists you can enter an optional *Description* here.

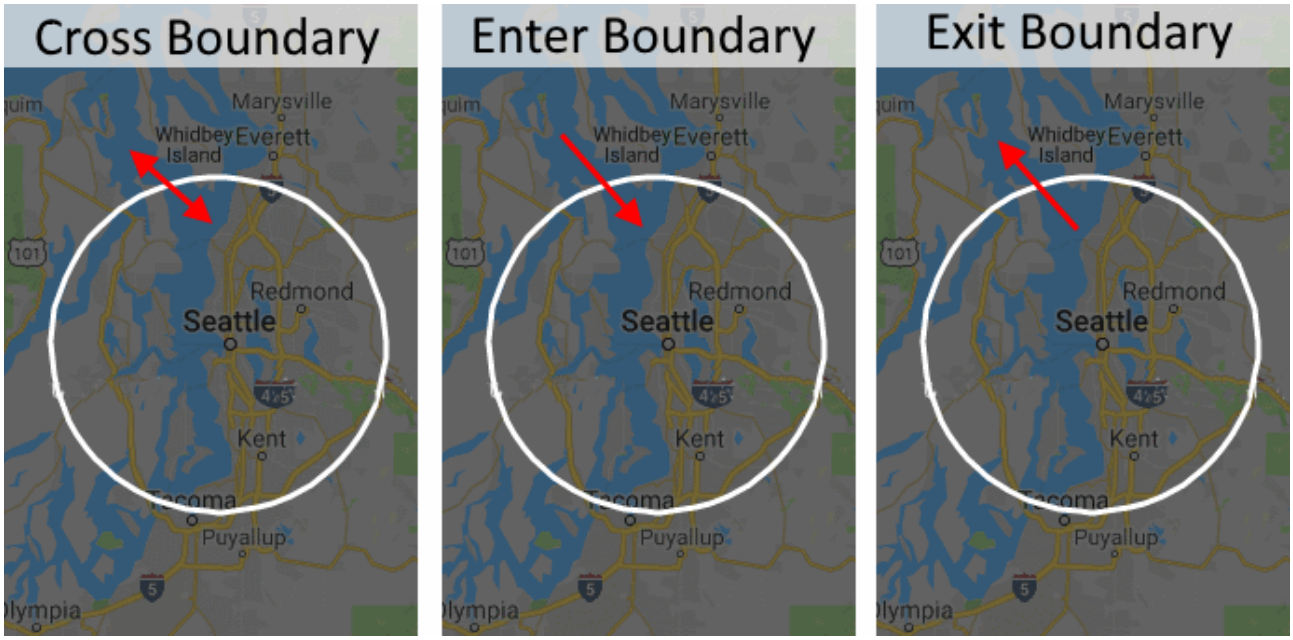
GIS Triggers are evaluated against when a ship crosses a boundary. Imagine a circle drawn with a radius of 'x' nautical miles from a point. That point could be a specific port, any known port or a custom user define point (determined by the type of trigger you create).

For the sake of an example we will say the point of reference is the port of Seattle and your radius is 20.00nm. Now that you have that circle with a 20nm radius around the port of Seattle. The ship enters the boundary when it goes INTO the circle / arrives in Seattle. The ship exits the boundary when it leaves the circle / departs from Seattle.

You can define as many rules as you like, each operates independent of the others.

⚠ Be sure to click Save and don't forget to Enable the trigger if you want it to be active!

Boundary Examples - 20 nautical miles from the port of Seattle



Arrive/Depart Any Port

These are triggered when arriving or departing any port in the SCS database.

+ Add a "Arrive/Depart Any Port" trigger

Name	Enabled	Description	
No triggers defined for this type.			

Arrive/Depart Specific Port

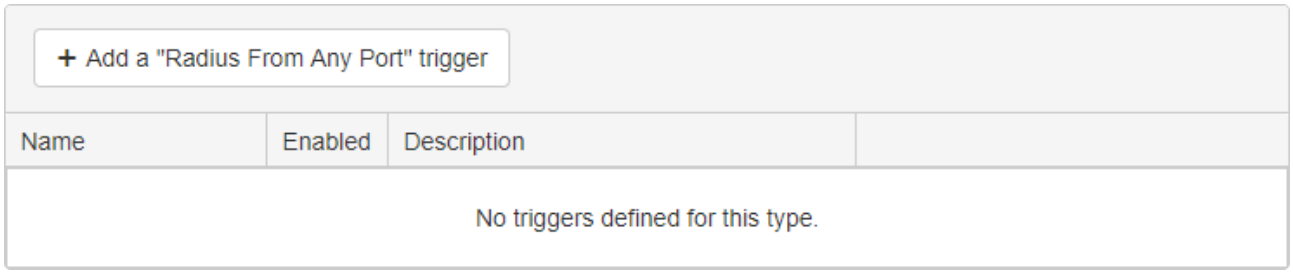
These are triggered when arriving or departing a specific port in the SCS database.

+ Add a "Arrive/Depart Specific Port" trigger

Name	Enabled	Description	
No triggers defined for this type.			

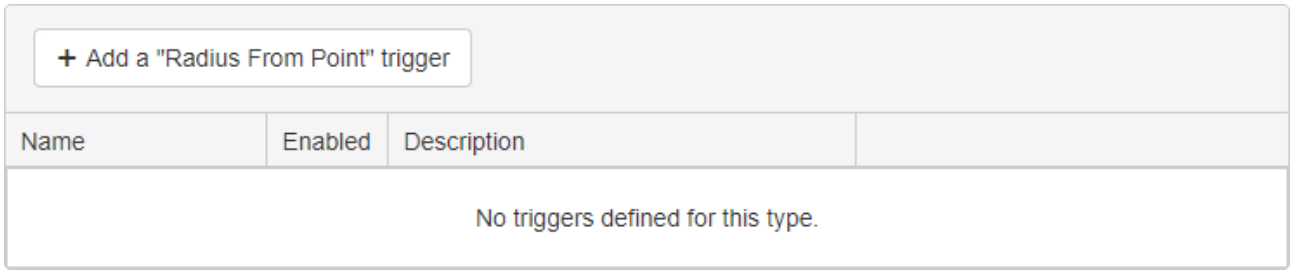
Radius From Any Port

These are triggered when the vessel enters or exits an area defined by a radius from any port in the SCS database.



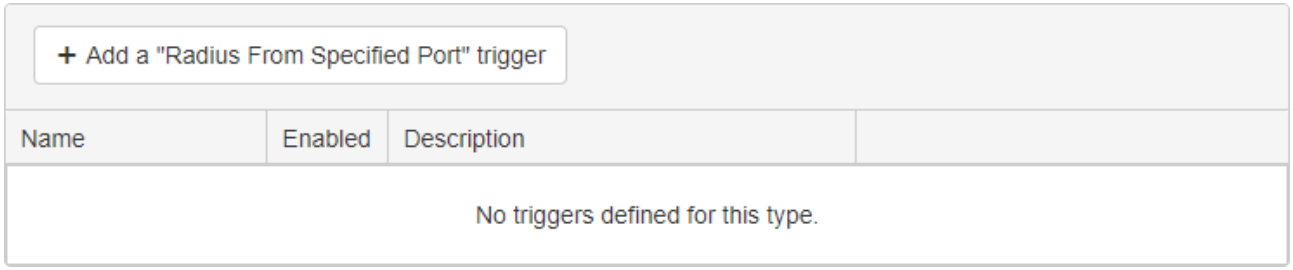
Radius From Point

These are triggered when the vessel enters or exits an area defined by a radius from the specified user defined geographic point. In the UI builder you will have to specify a single Latitude / Longitude coordinate from which to base these triggers off of.



Radius From Specific Port

These are triggered when the vessel enters or exits an area defined by a radius from a specified port. Each trigger is associated with a single port. In the UI builder you will have to select a single port from which to base these triggers off of.



⚠ GIS Triggers are actively monitored via the GIS Trigger Service which runs in the background. This service as well as the data acquisition service must be running in order for the triggers to work.

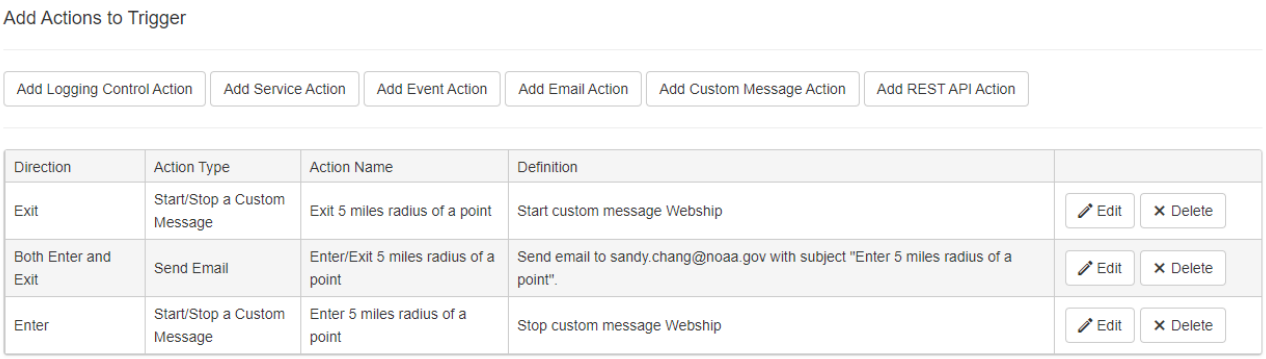
⚠ GIS Triggers utilize the SCS reference GNSS device, you must have at least 1 source providing the ships Latitude and Longitude for the triggers to work.

To add a port that is missing please contact us and we will add it.

Associate Actions

While you use the above portion of the builder to define WHEN a trigger should fire, you use this portion to define WHAT happens when the trigger fires. There are a variety of different Actions you can manage.

⚠ Be aware that changing or deleting any Action will also impact any trigger referencing it!



The main set of Action types allow you to perform a variety of different things such as Starting or Stopping windows services (for instance start or stop SCS data Acquisition), SCS Events or SCS Custom Messages. You are also able to send emails off the ship to notify shore-based personnel or trigger shore-based tools.

Each action allows allows you to specify which direction (entering or exiting the boundary defined by the trigger) the action should fire. This means in the Seattle trigger example above you can have different actions fire when you are coming into Seattle (eg email shore and shutdown all Events) vs leaving Seattle (restart all Events).

Add Logging Control Action

This action allows you to start or stop ACQ logging itself. Note you cannot (should not) stop the service, however this will prevent data from making it into the files and database. Data collection and dissemination (visualizations, other clients, custom messages, etc) will continue unabated.

Add Service Action

Allows you to start or stop windows services.

⚠ Do NOT stop ACQ with this action unless you are aware of the implications. If you stop ACQ then data acquisition will also stop and future triggers (such as 'restart ACQ when we leave port') will never fire! Use the *Add Logging Control* action to control ACQ logging instead.

Add Event Action

Allows you to start or stop SCS Events. Be sure to have sequences tied to the Event Start/Stop auto triggers if you want things to automatically start/stop inside the event itself.

Add Email Action

Allows you to send an email to a party on or off the ship.

Add Custom Message Action

Allows you to start or stop Custom Messages (send data to other programs, etc).

Add REST API Action

Allows you to call remote RESTFul programming interface actions. More information can be found in the GIS Trigger API Action section of this documentation.

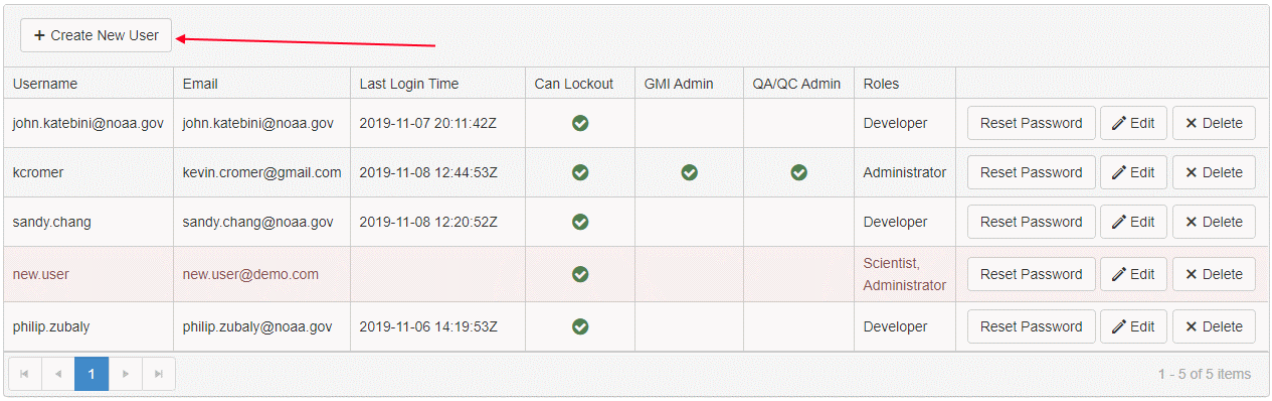
User Management

SCS does not require a login for all of it's functions, for instance if templates are shared appropriately then users can run them without having to log in. However, many operations inside SCS require the user to be authenticated before allowing them to be performed. Any action which is tied to an account, such as creating or editing templates/widgets/layouts require authentication. Building Custom Messages, participating in Events, etc all require the user to have an active account before they can proceed.

Users cannot self-register, an administrator must approve and create them an account. This is done via the *Manage Users* link under the *System* menu item.

The main grid presents the list of users currently registered to use your local SCS system. You can click on the column headers to sort by their content (so you can quickly see which accounts haven't been logged into in a while, which have admin rights, etc).

Users who haven't logged in will be highlighted in red such as "new.user" below

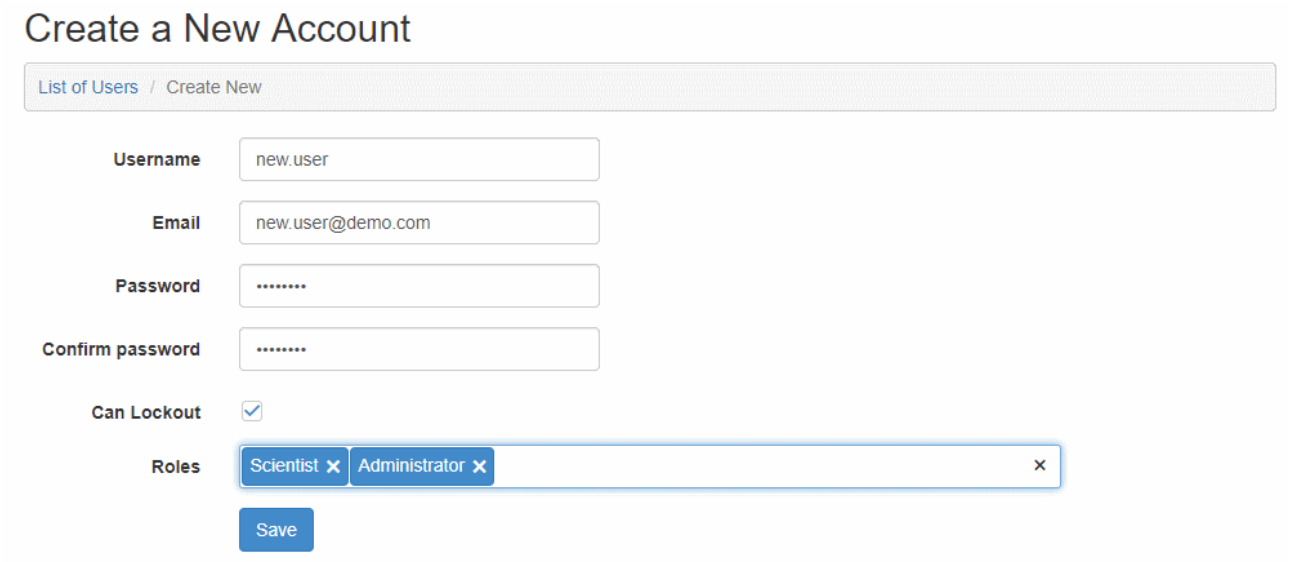


To create a new user simply click the *+Create New User* button located in the upper left corner of the grid.

If you want the account to be automatically locked out of SCS for security reasons (such as if they enter the wrong password to many times in a row) be sure to check the *Can Lockout* checkbox.

There are two main roles available to new users, very few will be Administrators, most will be scientists.

> ⚠ If you grant admin rights to the new account be aware it will have the same rights you currently do and full admin access to all of the system.

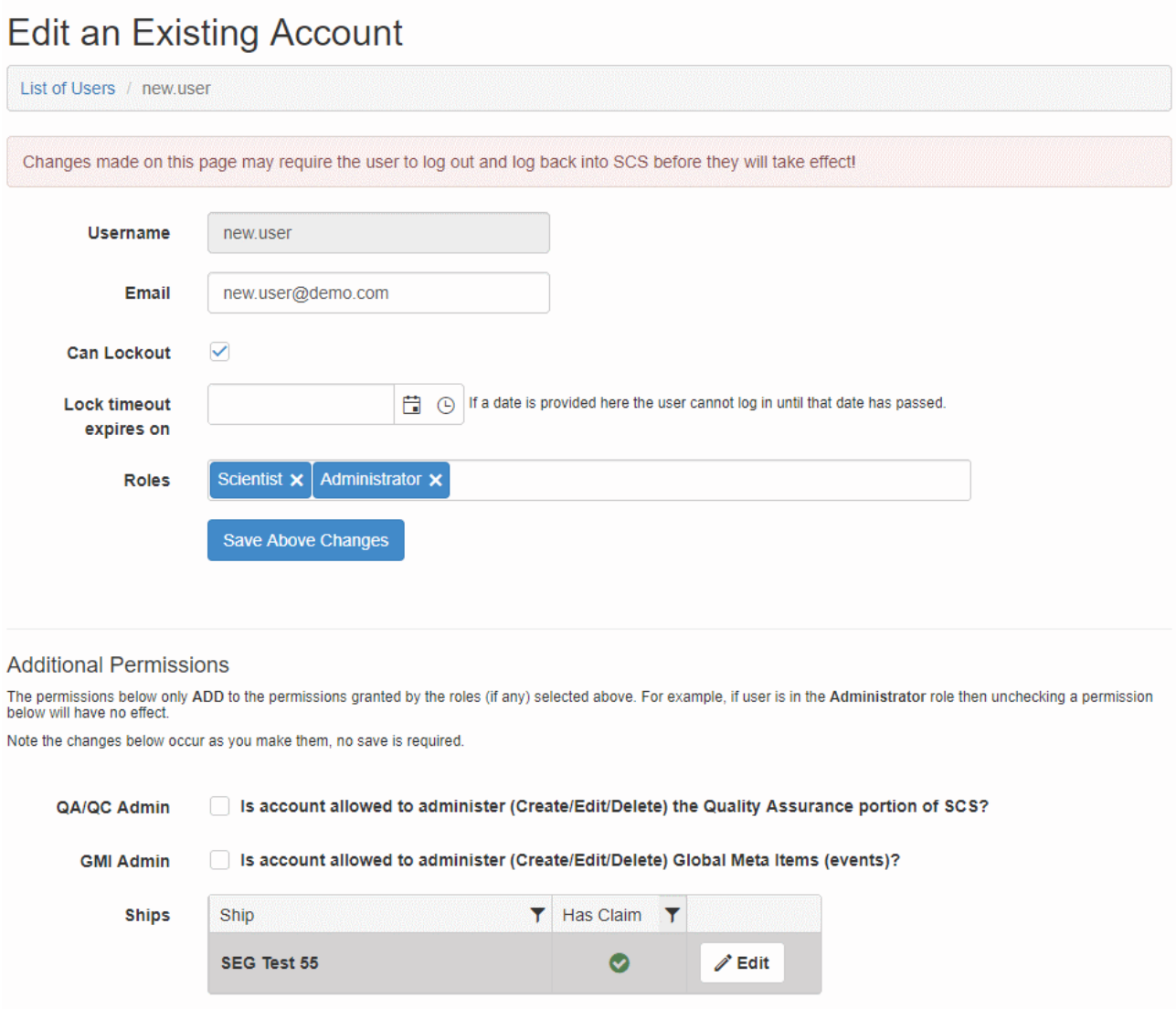


On the main grid, at the end of each row is a set of buttons which allow you to perform actions on the given user's account. Some of the changes might require the user to log out and then log back in before they will take effect.

> ⚠ If you delete an account all templates assigned to it will be re-assigned to you. Deleting an account cannot be undone!

When editing a user you may notice you cannot change some of the things initially specified when creating their account. You'll also notice you can specify some additional fine-grain privileges such as granting a non-admin account semi-admin rights in certain subsections of SCS. For instance you can allow an account in the Scientist role to have full admin rights in the QA/QC subsystem inside SCS. This user can not do general

admin duties, such as modify CFE or create new users, but they CAN now create, update and delete QA/QC definitions which a normal scientist account can not do.



Vessel Profile

The Vessel Profile link found under the System menu allows administrators to maintain the metadata regarding their ship and the personnel operating her. The data entry is organized into multiple tabs which are described below.

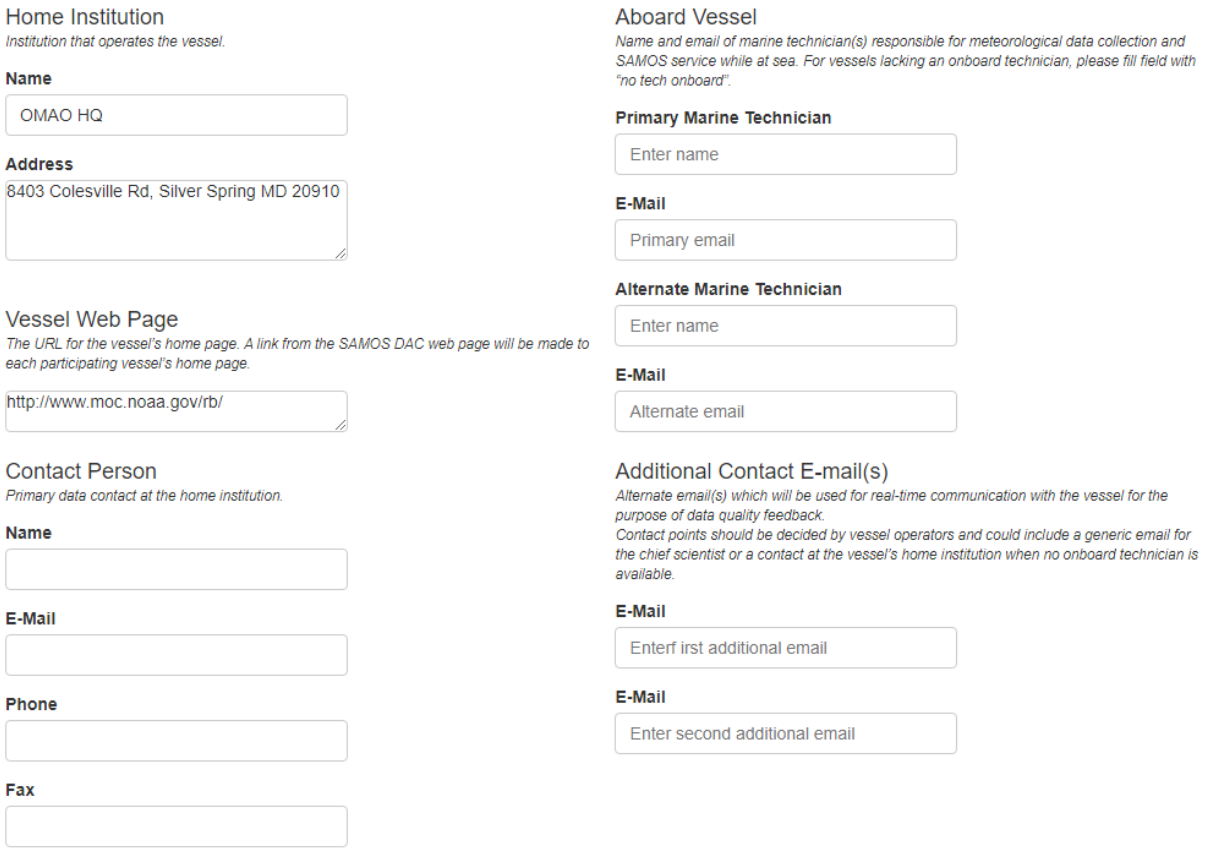
Basic Information

The Basic Information tab allows you to enter some of the core metadata used to describe your vessel. Some of this data is sent with the data packages sent to shore so be sure to keep the information as accurate as possible.

The *Recruiting Country* and *Date of Recruitment* elements are specific to the SAMOS program. If you are not enrolled in SAMOS you can ignore these.

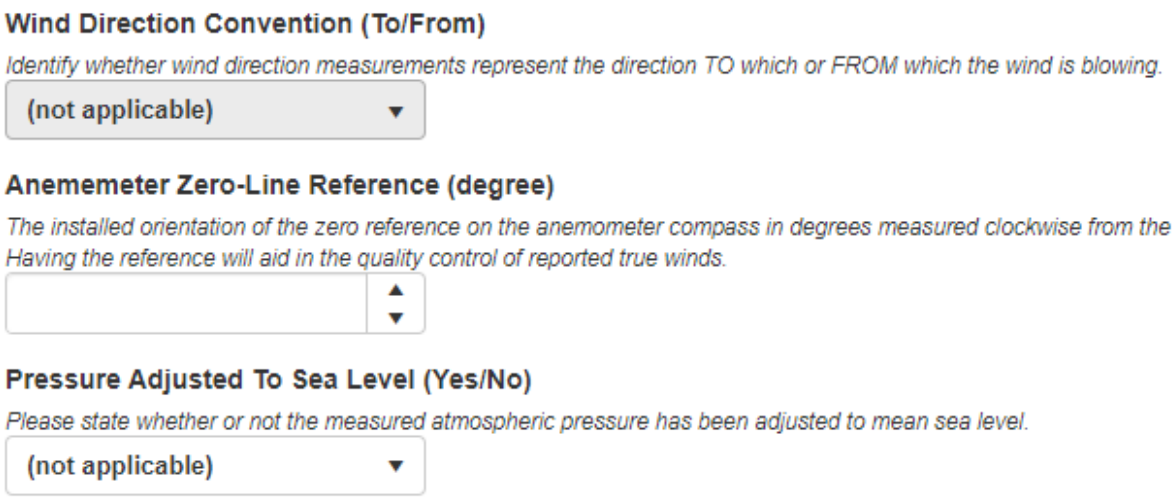
Contact Information

The *Contact Information* tab allows you to enter the information used by others to contact your vessel in the event of questions or issues regarding sent data or any other reason. The email addresses specified here are sent along with most data packages sent to shore such as NCEI and SAMOS.



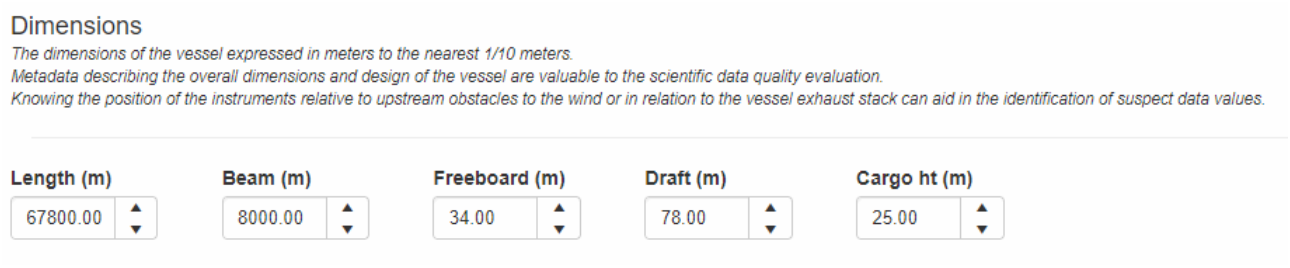
Device Metadata

While most metadata regarding specific physical devices is completed inside CFE there are a few (SAMOS requested) metadata items which span beyond any single device. Those can be clarified via the *Device Metadata* tab. Details on what these items mean can be found on the SAMOS main website.



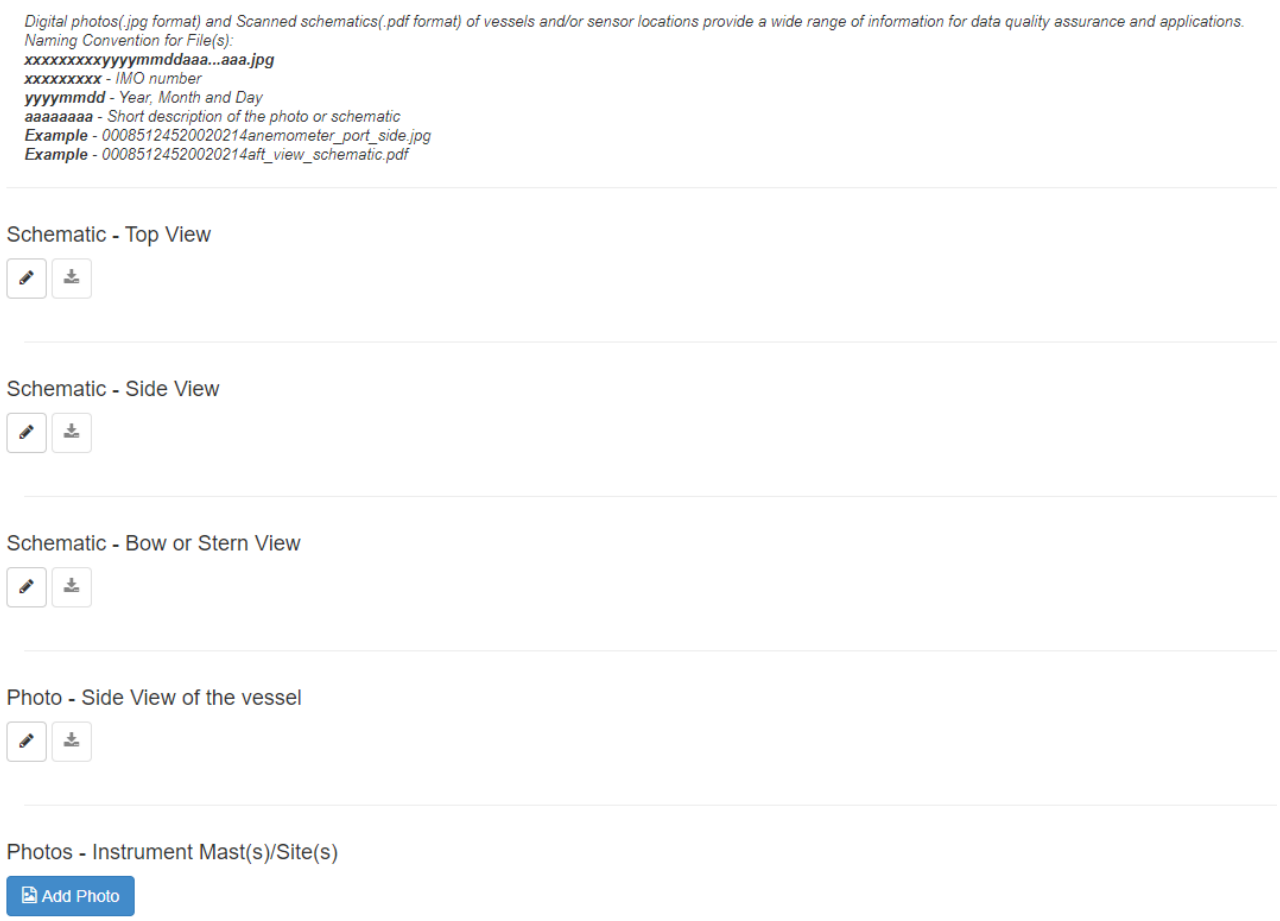
Vessel Layout

The Vessel Layout tab allows you to enter the dimensions of your ship, all measurements should be in meters.



Photographs and Schematics

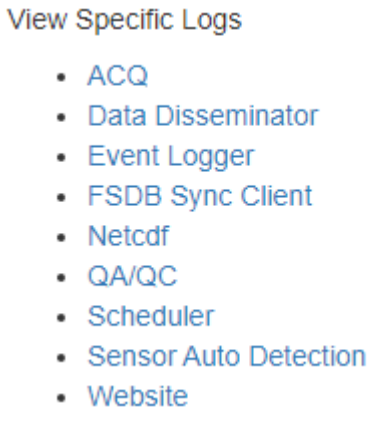
If you have images of your vessel from the various viewpoints and angles detailed in this tab please upload them. If you have schematics (CAD diagrams or printouts, etc) you can also provide them via this tab. Click the appropriate edit button to begin the upload process.



Diagnostics Logs

System logs can be reviewed in the *Logs* page located in the *System* main menu item. Each logged row will have the timestamp it was sent, the source that sent it, one of a few 'log levels' which define its criticality and the log message itself. In the event things aren't performing as expected or as part of your normal maintenance routine it is advisable to check these logs to keep tabs on the underlying SCS system.

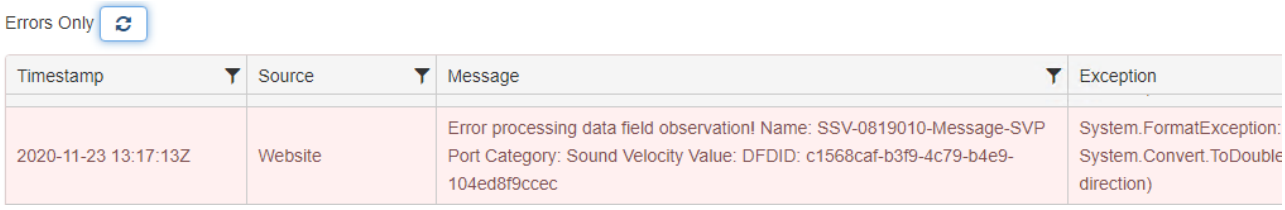
On the root page you will see links to filter down to specific log sources, click on a link to scope your review to that single source (for instance if you are only interested in seeing log messages generated from the Event Logger service).



You can also clear all the logs via the *Delete all logs* button located in the upper right corner of the screen.

Below the links you will see two grids, the first grid is a running log from all sources no matter what the log level. The second grid is similar, however it only shows messages with the log level set to Error or worse. In addition the second grid provides a column with a potential exception message and/or call stack to help you resolve the issue that was encountered.

Unlike other portions of SCS this page does not automatically refresh itself. If you want to refresh your view you can either refresh the entire page or click the refresh button located on the top of each grid to update that grid alone.



SignalR Hub Monitoring

SCS has a set of SignalR Hubs it uses to transmit signals and data back and forth among the various applications. Two important ones are the Configuration hub and the Data hub. The Configuration hub is used to announce changes to the sensor configuration inside SCS alerting templates and remote applications of changes made inside CFE. The Data hub is used to send sensor data to all the remote client websites & widgets directly from ACQ.

If you suspect an issue has arisen in either hub there are two pages located under the *System* menu item which allow you to passively monitor the hubs for traffic.

Monitor Data

The *Monitor Data* link will connect you to the Data hub allowing you to see the various messages being transmitted to all the clients. Data is sent in the JSON format, every second or so you should see new data items appearing here.

Monitor Configuration

The *Monitor Configuration* link will connect you to the Configuration hub allowing you to see publishes made via CFE. There are a lot of changes that can be made inside CFE and they have varying effects on the system overall. For instance, changing the comment on a *physical device* really doesn't make any different elsewhere in the system while changing the baud rate on a COM port most definitely would. The config hub breaks the changes up into categories and levels of impact. When you have the *Monitor Configuration* page open any publishes made by CFE should show up on the screen with the details.

Creating Support Tickets

If you encounter a bug, want to request a new feature or improve an existing one feel free to use the form located via the *Submit Ticket* link under the *System* menu item to send the information to us. Feedback provided this way automatically generates support tickets in our shore based project management system. As such it requires an active internet connection in order to work. If possible please include your email address when you submit the ticket so we have a way to communicate with you once we receive the ticket.

You can also reach out to us directly via the Contact page which provides a variety of tradition contact methods (phone, email, etc).

Create a Ticket

If you encounter a bug, want to request a new feature or improve an existing one feel free to use this form to send the information to us.

Note: This submission will travel over the internet, if you do not currently have a stable internet connection please wait until you do before attempting this operation!

Type of Ticket Bug 

Summary *

Description *

Email

Enter your email address so we can reach out to you for clarifications (optional but would be helpful)

Submit Ticket

* Required Field

SCS Events offer the ability to call remote (your) APIs via the Execute Web Hook action in any Action Sequence you define. This can be anytime during an event, such as when a button is pressed (CTD in water, net at depth, buoy deployed, etc) or a more general concept (when an event is over and ready to be processed).

This will always be via an HTTP POST.

The POST will have form encoded content (name/value pairs) consisting of:

SCS Version: The version of SCS which is making the call

TemplateId: The unique ID of the Event Template which was used to define the event making this call

EventId: The unique ID of the Event instance making this call (can be used later to retrieve the data for this event via the GetEventData API endpoint)

EventName: The name of the event making this call

EventStart: The time stamp of when this event started (ISO 8601 compliant format such as 2008-06-15T21:15:07.0000000)

ActionExecuted: The time stamp of when this event action was triggered (ISO 8601 compliant format such as 2008-06-15T21:15:07.0000000)

When a user configures the Event template they decide if any actual event or sensor data is sent over in this call. If they decide to include this then each item will also be included in the form data above POST'd to your endpoint.

SCS GIS Triggers offer the ability to call remote (your) APIs when the ship has GIS related events.

This may result in an HTTP GET or an HTTP POST depending on how you have your trigger configured.

If the trigger is based around a known port then information regarding that port is serialized into a JSON object and stored in the x-scs-port custom header sent to your code.

SCS provides a very basic REST API for users to call remotely via their own applications. You can access this API via HTTP GET calls to the primary SCS web server with the /SVC route.

> If you have a need or suggestion to improve this functionality for your usage please reach out. As this is a more advanced feature of SCS we have not built out to robust an interface as we are not as cognizant of the user-bases requirements so we're very willing to tailor it once we know what exactly you need.

> By default all methods below are HTTP GET, any deviation will be otherwise noted for the given endpoint.

GetCurrentDeviceConfiguration

[SCS HOST]/svc/GetCurrentDeviceConfiguration

Returns the current device configuration XML for the ship

GetDeviceConfiguration

[SCS HOST]/svc/GetDeviceConfiguration/{xmlReleaseID}

Returns a specific device configuration XML for the ship based upon the passed xmlReleaseID parameter

GetMessageDefinitionForDataField

[SCS HOST]/svc/GetMessageDefinitionForDataField/{xmlReleaseID}/{datafieldID}

Returns the configuration XML for a single Message Definition which contains a specific datafield.

xmlReleaseID parameter will specific which Device Configuration to search

datafieldID parameter will search the Device Configuration for the single specific Data Field who's parent Message Definition will be returned.

GetAcqLoggingStatus

[SCS HOST]/svc/GetAcqLoggingStatus

Returns JSON informing caller whether or not ACQ is currently logging or not.

> ⚠ This does NOT indicate if ACQ service itself is running or not. ACQ may be actively sending out data to clients even if it is not logging that data to disk!

JSON will have a success boolean, a state string and optionally an errorMsg string if an error is encountered.

```
{
    "success": true,
    "state": "Started"
}
```

IsRunningLatestVersionOfSCS

[SCS HOST]/svc/IsRunningLatestVersionOfSCS

Returns a JSON result informing caller if local ship installation of SCS matches what is currently available from shore.

JSON will have a single isUpToDate boolean.

```
{
    "isUpToDate": false
}
```

GetEventData

[SCS HOST]/svc/GetEventData/{eventIdStr}/{format?}

Returns data collected by an SCS Event in the requested format.

> The eventIdStr parameter can be obtained by the event itself. The conceptualized flow here is that the event calls your code via an Execute Web Hook action sequence (which provides your code with the eventID string used here) when appropriate (such as when the event ends so you can automatically and instantly download the event data).

eventIdStr parameter is the ID of the event run you wish to query for data

format parameter is optional, default value is "xml". Valid values are "xml" or "legacy"

SCS also offers shore-based API endpoints. These are cross-ship capable and are further defined at https://scsshore.noaa.gov/svc/API