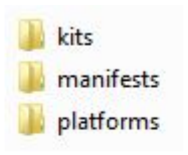
Understanding Sedona Kits and Kits Manifests

According to the SedonaDev.org site, a kit is the basic unit of modularity in the Sedona Framework. Kits contain one or more components and it is the components that are assembled onto a wire sheet and linked together to form applications. If a system integrator wants to use a particular component in an application, that component must be part of a kit installed on the target Sedona device. If the kit is missing, in most instances it is possible to load that kit into the Sedona device using the Kit Manager found in a Sedona tool such as Niagara Workbench or Sedona Application Editor (SAE).

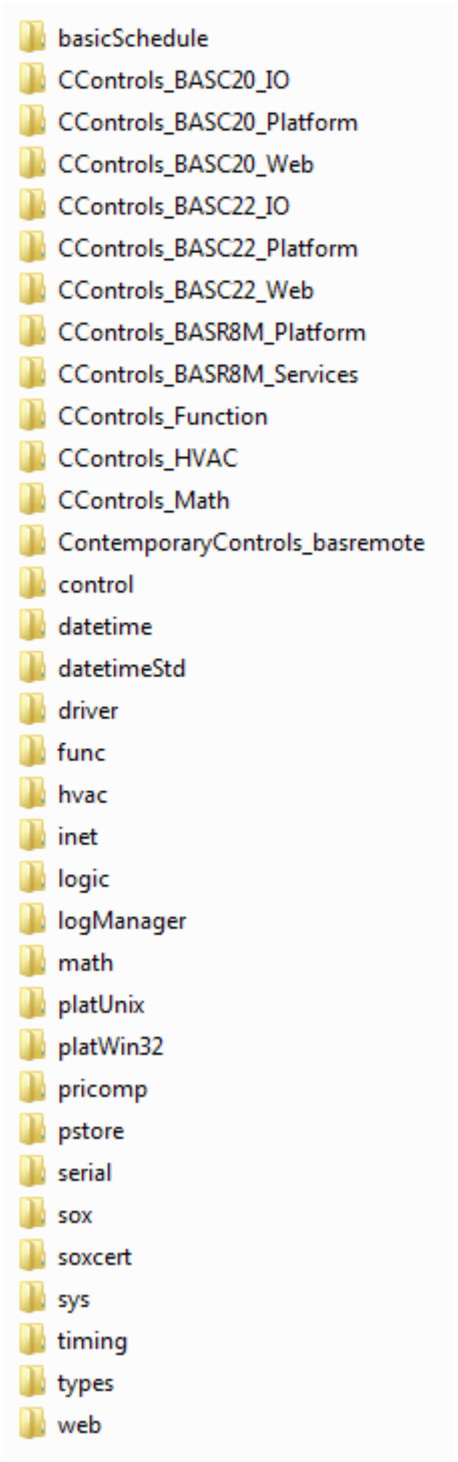
Modularity of kits is important if the Sedona community is to share components among its members. Contemporary Controls is a Sedona community member and offers its hardware-independent kits to Sedona community members in the form of a component bundle. This bundle is to be installed in the member's Sedona tool. This document attempts to explain the concept of kits and kit manifests as they relate to the sharing of hardware-independent components.

Component Bundle Structure

A component bundle is the term used to identify a single zipped file consisting of kit, manifest and platform files relevant to a particular vendor's Sedona device or devices. It is identified by a Sedona vendor's name along with a version identifier and it is provided by the Sedona vendor so that Sedona tools such as Niagara Workbench or Sedona Application Editor can be used to program the vendors' Sedona devices. In the case of Contemporary Controls (CC), CC has chosen to include information on all its Sedona devices with each release of a single component bundle such as Component_Bundle_BASC_1.0.40. If there is a change to a component within a kit requiring a kit change, CC will append the new kit and manifest files to the component bundle and release the bundle with a change in revision number. Kit and manifest files are not to be deleted – new ones are always appended.



If you extract the component bundle onto your desktop you would see one folder called sedona. This would be what is called sedona home. Expanding this folder will yield just three folders – kits, manifests and platforms.



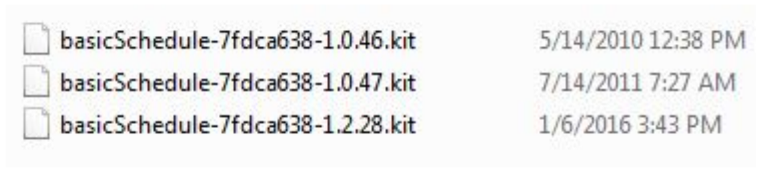
In terms of portability, the platform folder, or what is called the platform database, can be ignored because platform information is unique to each Sedona vendor and dependent upon native implementations. It is not that this information is unimportant; it is just meaningless to share among community members. What is more of interest to the Sedona community members are the kits and the kit manifests that describe the kits. The kits folder is called the kits database and the manifest folder is called the manifest database. If you expand either the kits or manifests folders, you will see what appears as identical structure of folders – each one devoted to a particular kit. However, the function of the two databases is different. We will first examine the Component_Bundle_BASC_1.0.40.

Kits can be characterized as being Tridium-release, custom hardware-independent, and custom hardware-dependent. The Tridium-release and custom hardware-independent kits can be shared with members of the Sedona community for use with their Sedona implementations. The custom hardware-dependent kits rely upon native hardware implementations and are usually useless to share. You can identify the kit type as follows:

- Tridium-release kits have no vendor name – the vendor is assumed Tridium
- The custom hardware-independent kits have the Sedona vendor name
- The custom hardware-dependent kits have the Sedona vendor name and model reference

Of the custom kits shown on the right, only the CControls_Function, CControls_Math and CControls_HVAC are custom hardware-independent kits and can be shared with the Sedona community. Do not assume that all Sedona devices incorporate all the kits listed – they do not. For example, the BASC20 uses the BASC20 hardware dependent kits and not the BASC22 or BASR8M kits. Not all Sedona devices use the Tridium control kit which was an earlier version that included almost all of the kits on the list. Tridium later broke up the control kit into several kits with more meaningful kit descriptions such as Math, Func and HVAC. Do not confuse the CControls_Math and CControls_HVAC kits with those of Tridium (Math, HVAC). Contemporary Controls created custom components that complement the Tridium components but elected not to merge these components with the Tridium-release components. Contemporary Controls' policy was not to modify any of the Tridium components unless absolutely necessary in creating Sedona virtual machines, or add components to a Tridum kit. All Contemporary Controls' kits have a CControls prefix.

Kits and Manifest Naming



basicSchedule-7fdca638-1.0.46.kit	5/14/2010 12:38 PM
basicSchedule-7fdca638-1.0.47.kit	7/14/2011 7:27 AM
basicSchedule-7fdca638-1.2.28.kit	1/6/2016 3:43 PM

We will study how the kit and manifest files are named by expanding the basicSchedule kit which is a Tridium-release kit.

In this instance there are three versions of the basicSchedule kit that are available for use. Each version is identified by kit name, checksum and revision. Up to a four-digit revision number is allowed of the format: major.minor.build [.patch]. Kits must be globally unique. That is why Tridium proposed naming rules for vendors so no naming conflicts occur. During creation of the kits, the checksum is automatically calculated based upon component types within the kit and the number of slots used by the components. However, the revision level is supplied by the developer of the kit. The operation of a component within a kit could be changed with modified code but that may not necessarily change the checksum of the kit. That is when a revision level needs to be incremented by the developer.

In the dateSchedule example provided, a Sedona device can only use one of the three possible kit versions listed or none at all. During Sedona development Tridium released several Sedona builds beginning with Sedona -1.0.31 on up to the current build Sedona 1.2.28. Kits could have incurred modification during the development process which accounts for the different revision levels. Other kits only have one revision level. Generally speaking, the later version is the one to use but older Sedona devices might rely upon earlier versions so that is why it is best to retain earlier versions of the kits in the component bundle.

The kit files (.kit) are not human readable. They are a compact representation of their components so they can be installed in embedded Sedona devices with limited memory. For example, the binary format of a Sedona application (app.sab) does not contain kit, type, or slot definitions, only their numeric ids. So in order to reconstruct the original application from its binary SAB file, we must have some external way to map those ids to the right definitions. Another file is needed called the kit manifest or simply the manifest file. Notice we have a one-to-one matching of a kit folder to a manifest folder. Next we will look at the basicSchedule manifest folder.

There happens to be only one file in this folder. The naming is similar in that this file has the same checksum but there is no version number. However, this file is an XML file (.xml) and is human readable but tedious to read. The checksum in the manifest folder must match the checksum in the kits folder. With our basicSchedule example, the single manifest entry will work with all three kit versions because the checksums are all the same.

Sedona Tool Database

A Sedona tool such as Niagara Workbench or Sedona Application Editor that is to connect to a Sedona device must have the same kits and manifests in its database as the ones used on the attached Sedona device or an error message will occur prompting the user to install the missing kits or manifests. Therefore, it is up to the Sedona vendor to provide the latest kits and manifest information conveniently available to the Sedona community especially for newly released products. One way of doing this is with a Component Bundle.

Installing Component Bundle to Niagara Workbench

Assuming Sedona is installed on a Niagara Workbench 3.7 or 3.8, download the latest component bundle from Contemporary Controls web site and place it on your desktop or to another convenient location. Open up Workbench and go to Tools>Sedona Installer. Make sure the Install TXS Bundle is checked and then browse for the bundle which should remain zipped. Chick Next, then Finish, and the new bundle will be installed. Workbench will write over all files of the same name but leave all other files intact.

Installing Component Bundle to BASbackup

Download the latest component bundle from the Contemporary Controls' web site and place it on your desktop or to another convenient location. Find the location of the current component bundle in the BASbackup directory. Either rename it as old or delete it. Copy the new bundle to the BASbackup directory while leaving it zipped.

Installing Component Bundle to Sedona Application Editor (SAE)

Download the latest component bundle from the Contemporary Controls' web site and place it on your desktop or to another convenient location. Find the location of the ccontrols data folder in the SAE directory. Extract the component bundle to this location. If you receive messages, select merge.

Using the Kit Managers

Once the bundle is installed in the Sedona tool, the kit manager within the tool can be used to add or remove kits. For the Sedona Application Editor, video 7 in the SAE series: Using the Kit Manager can be viewed to understand its operation.