Part 83 is very limited in its description of the contents of this [*Descriptor*] file

My first concern is regarding what a controller specifies and how it changes after a program is loaded. I think Brian asked a similar question regarding workflow.

* Multiple versions of the controller’s Descriptor file may be stored in the *AutomationComponent*. The *Descriptors* component of the AutomationComponentType is a folder that may hold multiple references to *Descriptors* (*AcDescriptorType*). This allows for *Descriptors* of this *AutomationComponent* to be added as they are modified throughout the engineering workflow, and separate *Descriptors* may be added for *FunctionalEntities* and *Assets* to support various sorts of system architecture and workflows.
* Each new version of a controller’s *Descriptor* is clearly identified. A *Descriptor* must contain exactly one *Descriptor* manifest which is an XML file that defines the metadata for a *Descriptor*, including the URI of the *Descriptor* (*DescriptorIdentifier*), the *Descriptor* version (*DescriptorVersion*), and the version of the OPC UA FX standard taken from the nodeset file that the *Descriptor* adheres to (*OpcUaFxVersion*).

My second concern is not related C2C MVS. I am wondering if this file should be associated with functional entities and not the AutomationComponent. When we start discussing devices that have multiple functional entities, it is not clear to me how you can create 1 offline descriptor file to represent all the possibilities and combinations of functionality.

* *Descriptors* may include multiple information model files (AML file based on the OPC UA FX Library), each describing a separate module.
* Separate information model files can be tied together using internal references.
* Various combinations of information model files may be used to describe modular *AutomationComponents* and product families.

Note: discussion also included the identifier and version related to descriptor and also the content of the descriptor file. – also ProductInstanceURI is in the asset – should there be a relationship between them (maybe in some cases – device model)
Requirement state that it is product descriptor file – which would need to be explain

* Information model files may include *AssetTypes* that include the *ProductInstanceURI* parameter.
* The *Descriptor* manifest which included the *DescriptorIdentifer* and *DescriptorVersion* binds this information to any *ProductInstanceURIs* that may be part of *AssetTypes* contained in information model files in the *Descriptor*.