Thoughts About Extending the Behavior of DefaultAccess

Pursuant to https://fedora-commons.org/jira/browse/FCREPO-400

There are four places in a service deployment object that a dataStream ID/wsdl message part name are indicated:

1. foxml:digitalObject / foxml:datastream[@ID="WSDL"] / foxml:datastreamVersion / foxml:xmlContent / wsdl:definitions / wsdl:message / wsdl:part@name
    a. Conditionality: Unclear
2. foxml:digitalObject / foxml:datastream[@ID="METHODMAP"] / foxml:datastreamVersion / foxml:xmlContent / fmm:MethodMap / fmm:Method / fmm:DatastreamInputParm@parmName
    a. Conditionality: apparently indicated by @required, although this element doesn't appear to be part of the assembly of dissemination URL's or sdep validation
3. foxml:digitalObject / foxml:datastream[@ID="DSINPUTSPEC"] / foxml:datastreamVersion / foxml:xmlContent / fbs:DSInputSpec / fbs:DSInput@wsdlMsgPartName
    a. Conditonality: Required if present, although @DSMin could imply a lack of requirement
    b. Note: This attribute value must match a DataStream ID in the relevant Digital Object
4. foxml:digitalObject / foxml:datastream[@ID="WSDL"] / foxml:datastreamVersion / foxml:xmlContent / wsdl:definitions / wsdl:binding / wsdl:operation / http:operation@location
    a. Conditionality: Required if present
    b. Note: This reference is solely in the form of URL substitution patterns

In the 3.1 release of DefaultAccess, the assembly of a dissemination URL:

1. Parses the DSInputSpec section, and throws an exception if any DSInputs have a @wsdlMessagePartName that does not correspond to a DataStream ID
2. Uses the @wsdlMessagePartName value in parentheses to establish a substitution pattern
3. Replaces all occurences of the substitution pattern with the URL_REF value(s) indicated in the DataStream IDs
4. Throws an exception if the resulting URL contains any more apparent substitution patterns in parentheses

It seems like it would be straightforward enough to write an Access module that simply didn't throw the exceptions in step 1 above, but that approach still seems to ignore a lot of information that may be encoded in the sdep. There is also some obscurity in the lack of a separation of the wsdl message part name and a bound datastream ID.

Maybe the responsibility could be divided up as:

- DSInput binds a datastream ID to the to a partName, via @pid. If @pid is absent, the partName will be used as the datastream ID, and the current digital object will be the reference object. If @pid begins with a slash, it is understood to indicate a datastream ID on the current object. If @pid is only a Fedora pid, the object with the given pid will be the reference object, and the partName will be used as the datastream ID. This could further be elaborated by allowing partName to be a messageName/partName structured value.
- MethodMap indicates the source of a named parm with the input element, and the name of the bound message part in the @parmName. If @required="false", the value of @defaultValue will stand in for a missing datastream value. If @required="true", a missing datastream will raise an error
- WSDL Message Part name is used to establish the substitution pattern in http operations. If the dissemination URL has a substitution pattern with no corresponding partName, an error is raised.

Thinking about this raises another potential issue under the 3.1 CMA: If an object can have multiple CModels, how should multiple matching SDeps be resolved when building a dissemination? DefaultAccess defers to the first SDep that the triple store returns, which is effectively an undefined behavior.