Support for renaming published dimensions

As there is currently no OGC method (that I am aware of) for dimension discovery, clients have to know what dimension names to use when interacting with a multi-dimensional OGC service.  Even in instances where dimension attribution could be used to infer the type of dimension, there is no way to differentiate the purposes of similar typed dimensions other than the dimension name until something resembling dimension discovery is standardized. For example,  a forecast coverage may have two time dimensions - one is the forecast model run time and one is forecast validity time.

The client has to "know" that the dimension "reference\_time" is the forecast model run time, and that "time" is the forecast validity time.  Furthermore, creators of software clients have to embed this special knowledge into client software.  As a result, the service owners must be able to advertise dimension names consistently, regardless of changes in underlying reader / data libraries in order to maintain client compatibility and conform to any organizational standard names or naming conventions.

I’d like to offer my current use case as an example:

Under normal circumstances, we use the FMRC capabilities provided by netCDF-java via featureCollection XML files to aggregate datasets with different runtimes (these datasets are themselves NCML aggregations of .nc files). When opening such a featureCollection in java, we get a dataset that that has a dimension variable named ‘run’ for the different run times, and everything works beautifully. However, I’m working on a project where we have a custom runtime dimension configured as “**reference\_time**” which is a requirement from our customer. When the netCDF plugin encounters the metadata configuration for the custom dimension in the coverage configuration it attempts to look up a “**reference\_time**” dimension in the netCDF dataset. In this case, the plugin will not locate the correct runtime dimensions and throws a null pointer exception in some WCS / WMS operations. The exceptions in the service occur because Geoserver assumes that coverages are configured a certain way.

I am proposing a design for Geoserver to allow the user to define what the custom dimension name is, thereby giving the implementers greater flexibility and enable a way to get past the null pointer exception as well as allowing Buisness level standards for naming conventions with their interfaces.

I envision a setting in the dimension tab when viewing a published layer which will map custom dimension names available in the dataset to a user defined string. This string would be stored in the coverage.xml under the custom dimension element as an advertised name, perhaps as an example: “**<publishedName>run</publishedName>**”. When Geoserver then computes the custom dimension (and here is where I’m not sure it should ultimately be done) for example, when determining the “dimensionName” (as used in [AbstractDefaultValueSelectonStrategy.java](https://github.com/geoserver/geoserver/blob/55c4740060eb2b8809ae935a510ac83eeb923313/src/wms/src/main/java/org/geoserver/wms/dimension/AbstractDefaultValueSelectionStrategy.java)) Geoserver will first check for the “<publishedName>” element and use it’s contents henceforth, otherwise default to the custom dimension name as usual.