OCPI Component Selection:

OCPI has the capability to select a named RCC component implementation from many implementations that may be available in the library based on an applications requirements. The application loads a worker using the following container method.

```
virtual Worker &createWorker(const char *file, const PValue *aProps,
                             const char *name, const char *impl,
                             const char *inst = NULL,
                             const PValue *wProps = NULL,
                             const PValue *wParams = NULL,
                             const PValue *selectCriteria = NULL,
                             const Connection *connections = NULL) = 0;
```

Argument 7 “selectCriteria” is a property list that can be used to influence the container to select a specific implementation of the worker from multiple implementations that may be available in the library. For this to function, a worker must provide a set of properties with default values that describe important characteristics pertaining to the implementation.

For example a FFT worker may advertise throughput and precision. An application that has a specific requirement for one or both properties can specify these requirements in the selectCriteria properties list and get the best fit FFT implementation available.

Format:

The optional selectCriteria property list is used by the container to select the best possible worker implementation based on the properties that are provided by the application.

The selectCriteria properties can be divided into two primary categories, selection variables and expressions. Type definition properties can be used in one or more expressions and are primary used as a convenient way to define a variable that can later be “tweaked” to fine tune the applications performance.

Selection Variable Properties:
Expression properties are evaluated as Double values, therefore the only valid type definition properties are Double’s.

Example:

```
PVDouble("Appthroughput", 1000000 );
```

Expression Properties:
An expression property is a single mathematical boolean expression that is evaluated based on one or more worker properties. The expression can also contain selection variable properties defined by the application. A boolean result is calculated from the

expression. Note that the expression is evaluated with each value converted to a Double so care must be taken when evaluating equality. All non-numeric values in an expression are presumed to be properties and are converted to the property values when the expression is evaluated. The property name search order is: worker properties first and then applications selection properties. If a property defined in an expression is not found it is assumed to be a mis-spelling and an exception is thrown. If the property is found but does not contain a default value, the expression is evaluated as false.

Example:
"Appthroughput < throughput"

Required Expression:
An application can specify a required expression using the following format:

```
PVString("__ocpi__exp-required", "throughput > Appthroughput");
```

A required expression implies that if the requirement is not met the implementation is not suitable and is eliminated as a candidate.

Scored Expression:
A scored expression is evaluated and if "true" the score is added to the implementations total. The total scores of all viable implementations are evaluated and the worker with the highest score is chosen as the best fit for the application. Negative scores are also valid.

An application can specify a scored property using the following format:

```
PVString("__ocpi__exp-scored 40", "distortion <= .001"),
```

In this example, if the workers property named "distortion" is less than or equal to .001 then the implementation will get a +40 added to its total score.

Expression Syntax:
The expression syntax is defined as follows:

lhs math expression <condition operator> rhs math expression

Both the left and right hand side expressions are "C" language math expressions that can include variables defined by the workers or the application. The following functions are also supported, sin(), cos(), tan(), atan(). The condition operator can be any one of the following, (==, <=, >=, !=, >, <).