HOW TO turn an Ascii file reader into a Linkable Component

How to create an AsciiReader

The input for model systems is often contained in simple ASCII files. Wrapping these files allows you to make the data available to OpenMI-compliant systems. For example, you can wrap the output file from one component so that it serves as the input for another component. When such a file is wrapped, it can be accessed in the same way as any other OpenMI component. The wrapped output file makes it easy to test the OpenMI component that uses the file as input. It will also be possible to run the component using the data without having to run the delivering component.

To be OpenMI-compliant, a component needs to implement two interfaces: the IPublisher and ILinkableComponent interfaces. Therefore the ASCII reader needs to implement these interfaces. As with model systems, this can be done by inheriting from Oatc.OpenMI.Sdk.Wrapper.LinkableEngine and implementing specific methods.

The difference between the wrapping of ASCII files and the approach used for model components lies in the implementation of the ExchangeItems (GetOutputExchangeItem) and the GetValues call. In the case of a model component, the ExchangeItems will usually be provided by a populated model kernel. When wrapping an ASCII file, the information about possible ExchangeItems must be contained in the ASCII file itself.
Figure 1 is an example of an ASCII file (get file; there are four lines of comments, followed by a line specifying the quantity and another containing the three locations.

```
// Lines starting with // are comment
// The first uncommented line defines the quantity
// The second uncommented line defines the locations
// All next lines consist of a timestamp and values of the quantity for each location
"Flow"
"Loc1";"Loc2";"Loc3"
"2005-01-01 00:00";"15.4";"18.2";"22.4"
"2005-01-01 03:00";"15.3";"18.5";"22.5"
"2005-01-01 06:00";"15.4";"18.9";"22.4"
"2005-01-01 09:00";"15.2";"19.0";"22.6"
"2005-01-01 12:00";"15.1";"18.8";"22.7"
"2005-01-01 15:00";"14.8";"18.6";"22.9"
"2005-01-01 18:00";"14.7";"18.4";"23.4"
"2005-01-01 21:00";"14.7";"18.2";"23.8"
"2005-01-02 00:00";"14.6";"18.2";"23.9"
"2005-01-02 03:00";"14.1";"18.1";"24.0"
"2005-01-02 06:00";"13.8";"18.2";"23.5"
"2005-01-02 09:00";"13.9";"18.0";"23.6"
"2005-01-02 12:00";"13.6";"17.8";"23.5"
"2005-01-02 15:00";"13.2";"17.3";"23.3"
"2005-01-02 18:00";"12.8";"17.4";"23.2"
"2005-01-02 21:00";"12.7";"17.2";"23.1"
"2005-01-03 00:00";"12.6";"17.2";"22.9"
"2005-01-03 03:00";"12.1";"16.8";"22.7"
"2005-01-03 06:00";"12.2";"16.6";"22.5"
"2005-01-03 09:00";"11.4";"16.5";"22.4"
"2005-01-03 12:00";"11.6";"16.4";"22.2"
"2005-01-03 15:00";"11.7";"16.3";"21.8"
"2005-01-03 18:00";"11.6";"16.0";"21.5"
"2005-01-03 21:00";"11.2";"15.8";"21.4"
"2005-01-04 00:00";"11.0";"15.6";"21.3"
```

Figure 1 An example ASCII file

This information defines the possible OutputExchangeItems.

The OutputExchangeItems could look like the following:

```
OutputExchangeItem[0] = { "Flow", "Loc1"}
OutputExchangeItem[1] = { "Flow", "Loc2"}
OutputExchangeItem[2] = { "Flow", "Loc3"}
```

Implementing an OpenMI-compatible linkable component that uses this ASCII file as input and makes the contents of the file available through OutputExchangeItems requires the following steps:

- Start developing the component by implementing org.OpenMI.Utilities.Wrapper.LinkableEngine.
- Write code for methods such as Initialize, Finalize, GetModelID and GetModelDescription.

- Write code for the SetValues, GetInputExchangeItem and GetInputExchangeItemCount methods; these methods are empty because the ASCII reader does not accept data.
- Write code for the GetOutputExchangeItem method; make sure the ASCII file is read, the ExchangeItems are extracted from the file and the OutputExchangeItems are defined.
- Write code for the GetValues method; this should extract the correct data for the requested timestamp/ExchangeItem combination from the ASCII file.

Prototype code for GetOutputExchangeItem is given in Figure 2.

```
public IOutputExchangeItem GetOutputExchangeItem(int outputExchangeItemIndex)
{
return _output;
}

private void ReadFile()
{
        StreamReader reader = new StreamReader(_inputFile);
        bool quantityRead = false;
        bool elementsRead = false;
        _timeStamps.Clear();
        _elementValues.Clear();

        string line;
        while ((line = reader.ReadLine()) != null)
        {
                if (line.StartsWith("//"))
                {
                        // ignore
                }
                else if (!quantityRead)
                {
                        _quantity = new Quantity(line.Trim(' ', '"'));
                        _output = new OutputExchangeItem();
                        _output.Quantity = _quantity;
                        quantityRead = true;
                }
                else if (!elementsRead)
                {
                        string[] elements = line.Split(';');
                        _elementSet = new ElementSet();
                        _elementSet.ID = "File Contents";

                        _output.ElementSet = _elementSet;

                        for (int i = 0; i < elements.Length; i++)
                        {
                                _elementSet.AddElement (new Element(elements[i].Trim('"')));
                        }
                        elementsRead = true;
                }
                else
                {
                        string[] values = line.Split(';');
                        DateTime timestamp = Convert.ToDateTime (values[0].Trim('"'), _culture);
                        double modifiedJulianDateTime = CalendarConverter.Gregorian2ModifiedJulian(timestamp);

                        double[] locationValues = new double[values.Length - 1];
                        for (int i = 1; i < values.Length; i++)
                        {
                                locationValues[i - 1] = Convert.ToDouble (values[i].Trim('"'), _culture);
                        }
                        _timeStamps.Add (modifiedJulianDateTime);
                        _elementValues.Add (locationValues);
                }
        }
}
```

Figure 2 Code for GetOutputExchangeItem

A prototype of the GetValues method is shown in Figure 3.

```
public IValueSet GetValues(ITime time, string linkID)
{
        ILink outgoingLink = (Link) _links[linkID];

        if (outgoingLink == null)
        {
                throw new Exception ("Unknown link or quantity");
        }

        if (time is TimeStamp)
        {
                TimeStamp timestamp = (TimeStamp) time;

                double[] results;
                for (int i = 0; i < _timeStamps.Count; i++)
                {
                        if ( (double)_timeStamps[i] + _delta > timestamp.ModifiedJulianDay)
                        {
                                results = (double[]) _elementValues[i];
                                return new ScalarSet(results);
                        }
                }

                throw new Exception ("No appropriate values found");
        }

        throw new Exception ("Time should be of type TimeStamp");
}
```

Figure 3 Example GetValues method