



Aspen Engineering Suite 2004.1

Aspen Simulator Interface

Programmer's Reference

Who Should Read this Guide

This document is designed to be used by programmers using Aspen Simulator Interface (ASI) to access data from simulators from Aspen Technology. It also provides an introduction to ASI to those who are unfamiliar with it.

Contents

INTRODUCING ASPEN SIMULATOR INTERFACE.....	5
In this Manual	5
Implementation	5
ASNBroker	6
IASNNode	6
IASNCache.....	6
Common Data Model	6
Simulation Control	7
1 ACCESSING AND CONTROLLING SIMULATIONS	8
Opening Simulations.....	8
Dealing with an Open Document	9
Traversing and Accessing Simulator Data	10
Node Attributes.....	10
Node Children.....	11
XML Dump	12
Notification	12
Working with Caches	13
IASNContainer	13
ASNCache	15
Solving	18
Messages	18
Solve Methods	19
Message Filtering	21
2 FUNCTION REFERENCE.....	22
Services	22
enum ServiceType.....	22
Messages and Errors	22
enum Filter	23
enum FilterDestination.....	23
enum MessageSeverity	23
Interface IASNMessageSupport	23
Class ASNMessageEventArgs : System.EventArgs	24

Creating Objects	24
Object ASNBroker	24
Object ASNDocument	25
Accessing an Object's Data	25
Interface IASNNode	26
enum AttributeType	27
Object ASNAttribute	27
Interface IASNContainer	27
object ASNCache : IList, IDisposable	28
Interface IASNNodeLink	30
Interface IASNContainerLink	30
Solving Objects	30
Interface IASNSMSolve	31
APPENDIX A STANDARD ATTRIBUTES	32
APPENDIX B PATH SPECIFICATION (BNF DESCRIPTION)	35
GENERAL INFORMATION	36
Copyright	36
TECHNICAL SUPPORT	38
Online Technical Support Center	38
Phone and E-mail	39

Introducing Aspen Simulator Interface

Aspen Simulator Interface (ASI) is designed to provide access to a simulator's data and control the execution of the simulator from outside the simulator. Although various methods are currently available to get data into and out of the Aspen simulators, they have several shortcomings:

- They are very specific to each simulator.
- They are generally not efficient enough for access across machine boundaries.
- They are generally not efficient enough to access large amounts of data even on the same machine.

ASI is designed to address these deficiencies by creating:

- A single interface specification for all simulator data access and control.
- A common data model to access common simulator data.
- The ability to define large groups of data to be transferred in a single call.

In this Manual

This introduction provides an overview of the ASI components.

Chapter 1 provides a detailed description of each major function of ASI, with examples.

Chapter 2 provides an interface reference guide to all ASI interfaces, objects, and methods.

The appendices list the standard attribute names and path specifications.

Implementation

ASI is implemented as a set of C# classes and interfaces. Classes are prefixed with ASN (ASI .NET) and interfaces are prefixed with IASN.

ASI consists of several components:

- A broker object which is used to open or access running simulations and obtain the ASI interface to the simulation,

- A hierarchy of node objects which is used to access the data within the simulation,
- A cache object which supports bulk data transfer into and out of simulators
- A common data model for data which multiple simulators support
- A simulation control interface

ASNBroker

The ASNBroker object can be created by any process. It will accept a filename of an Aspen Plus, HYSYS, or ACM simulation and return the ASNDocument interface to the simulation. If the simulation is running, it will connect to the running version. If not, it will start the simulation and connect to it. From the ASNDocument, you can obtain the top node in the simulation's data hierarchy.

The ASNBroker object can also return an IASNCache reference which contains connections to what has been copied to the Windows clipboard. This is an easy way to connect to data which is displayed on a simulator's forms.

In the current version (aspenONE 2004, Update 1), simulations can only be accessed on the local machine.

IASNNode

The hierarchy of nodes is represented by instances of the IASNNode interface. From an instance of the IASNNode, you can get to the node's children or attributes. The attributes contain data such as the node's value and unit of measure (if it is a leaf in the hierarchy), or the id (the name of the node). This information can be accessed or set from the attribute.

IASNCache

Nodes can be collected together into an ASNCache object. This object contains some or all of the nodes' attributes locally, so that data may be accessed quickly and conveniently. The cache can be updated (data copied from simulation to local copy), or stored (local copy to simulation) in a single call.

The cache can be created from data on the clipboard, by adding one or more IASNNodes, by combining other caches, or by saving a cache into a stream, and restoring it later.

Common Data Model

What nodes you will encounter in a simulator's ASI node hierarchy depend on the structure of the simulation. Therefore, to access data in different simulations, you will use the same interfaces (IASNNode, etc.), but different paths to get to the data.

We are beginning to build a common data model between Aspen Plus and HYSYS. This is accessible under the CCDM branch of each simulator's ASI

data hierarchy. Data under this branch can be accessed through the same path in each simulator.

Simulation Control

From the ASNDocument, you can access the simulation control interface, IASNSolve. This interface allows you to test the simulation for readiness to solve, solve the simulation (steady state), and filter and receive messages generated by the simulator.

1 Accessing and Controlling Simulations

This section contains detailed descriptions of how to access data and control simulations with ASI. Specifically, the features described are:

- Opening simulations
- Dealing with the open document
- Traversing and accessing simulator data
- Working with caches
- Solving

For each functionality, a detailed description of usage and examples are included.

Opening Simulations

Simulations are accessed through the ASNBroker object. Its constructor takes no arguments:

```
using Aspentech.ASI;  
ASNBroker broker = new ASNBroker();
```

A simulation can be opened by filename. A string specifying the application's version can be specified:

```
using Aspentech.ASI;  
ASNDocument apDoc = broker.CreateObjectFromFile("c:\\test\\pfdtut.bkp");  
ASNDocument apDoc2 = broker.CreateObjectFromFile("c:\\test\\pfdtut.apw");  
ASNDocument hsysDoc =  
    broker.CreateObjectFromFile("c:\\test\\tutor1.hsc");  
ASNDocument acmDoc = broker.CreateObjectFromFile("c:\\test\\test1.acmf");  
ASNDocument apl22Doc =  
    broker.CreateObjectFromFileVersion("c:\\test\\pfdtut.bkp", "12.2");  
ASNDocument hsys30Doc =  
    broker.CreateObjectFromFileVersion("c:\\test\\tutor1.hsc", "3.0");
```

The broker can also be used to get a collection of data (ASNCache) off the clipboard. The current version of the application will be used. The name of the ASI supporting application, its version, and the currently open filename is returned:

```
using Aspentech.ASI;
ASNDocument doc;
ASNCache cache = broker.CreateCacheFromClipboard(out doc);
```

See **Traversing and Accessing Simulator Data** for information on how to use an ASNCache.

Dealing with an Open Document

Once you obtain an ASNDocument, you can get the top node in the data hierarchy or the solver interface through the **GetService** method:

```
using Aspentech.ASI;
IASNNode node = apDoc.GetService(ServiceType.Node);
IASNSMSolve solver =
    apDoc.GetService(ServiceType.SequentialModularSolve);
```

The document can be saved using the **Save** and **SaveAs** methods:

```
using Aspentech.ASI;
apDoc.Save();
apDoc.SaveAs("copy");
```

Finally, a document can be closed with the **Close** method. This will close the document and release all underlying COM references for all references to this document. Do not try to use any nodes, solvers, or other objects which you may have obtained for a closed document.

```
using Aspentech.ASI;
apDoc.Close();
```

Traversing and Accessing Simulator Data

The basic object for traversing the simulation's data hierarchy is the node, accessed through the `IASNNode` interface. Data about the node is exposed as attributes. From a node you can obtain parent and child nodes in the hierarchy.

The top node in a simulation hierarchy is obtained from the `ASNDocument`:

```
using Aspentech.ASI;
IASNNode node = apDoc.GetService(ServiceType.Node);
```

Node Attributes

A node's attributes are accessed are exposed as a .NET `IDictionary` of `ASNAttribute` objects. Get the dictionary through the **`IASNNode.Attributes`** property. Then you can get an individual attribute by name. For example:

```
using Aspentech.ASI;
IDictionary dict = node.Attributes;
ASNAttribute attr = dict["value"];
    or:
ASNAttribute attr = node.Attributes["value"];
```

Attributes have a datatype as defined by their `Type` property. It may be any of the values of the **`AttributeType`** enum:

- Empty
- Integer
- Real
- String

The attribute value can be accessed (set or get) through the **`Value`** property. The `Value` property will cast to the proper datatype when assigned or set:

```
using Aspentech.ASI;
ASNAttribute attr = node.Attributes["uom"];
attr.Value = "F";
double temp = node.Attributes["value"].Value;
```

Each simulation may support as many attributes at any node as it wants. However, we have established a set of standard attributes which simulators should support if the functionality allows. That is, not all nodes support a

uom attribute, but if the node is dimensioned, then the attribute will be called uom. The standard attributes are listed in Appendix A.

Node Children

Through the `IASNNode.GetChild` method you can access another node in the simulation's data hierarchy. This can be an immediate child node by supplying a name as the argument, or a more distant relative by supplying a more complex path.

A path consists of one or more names (or "*" to match any name), separated by ".".

```
using Aspentech.ASI;  
IASNNode child = node.GetChild("B1.Temperature");
```

A name may be replaced or followed by a "#n", where n is a zero based integer, to indicate the nth element which matches:

```
using Aspentech.ASI;  
IASNNode child = node.GetChild("B1.Stage.#3.Temperature");
```

If a name contains special characters, it should be enclosed in quotes:

```
using Aspentech.ASI;  
IASNNode child = node.GetChild("S1.Flow.\"C=C\"");
```

A path can include one or more conditionals enclosed in "[]" following any name. The conditional is a path, followed by an optional attribute name enclosed in "()", followed by an optional "= string". If the attribute name is omitted, the value attribute is used. If the "= string" is omitted, the conditional is true if the path resolves to an existing node. "!", "|", and "&" can be used to string together more than one conditional:

```
using Aspentech.ASI;  
IASNNode child = node.GetChild("B1.X[Component=C1]&[Stage=2]");
```

A BNF description of the path specification is included as Appendix B.

XML Dump

The contents of a node and all of its descendents can be dumped into an XML string. The attributes which are requested are listed as an array:

```
using Aspentech.ASI;
string[] attrs = {"id", "value", "uom"};
int size = node.GetXMLSize(attrs.Length, attrs);
string xmlRepresentation = node.GetXML(attrs.Length, attrs);
```

Notification

IASNNodes will raise a **DataChanged** event when their value, any child value, or structure of children is changed. They raise a **Deleted** event when they are deleted.

Attach to a node's **DataChanged** event:

```
using Aspentech.ASI;
class nodeholder
{
    IASNNode myNode;
    public nodeholder(IASNNode node)
    {
        // Upon creation of this class instance,
        // attach notification request
        myNode = node;
        myNode.DataChanged += new EventHandler(node_DataChanged);
    }
    protected override void Dispose( bool disposing)
    {
        if (disposing)
            myNode.DataChanged -= new
                EventHandler(node_DataChanged);
        base.Dispose(disposing);
    }
    private void node_DataChanged(object sender, EventArgs args)
    {
        // this code executed when node fires DataChanged event
        // myNode and sender will be the same object
    }
}
```

Attach to a node's **Deleted** in a similar manner.

Working with Caches

Two classes support the efficient use of collections of data, the **IASNContainer** and the **ASNCache**. The **IASNContainer** is a dynamic container used for constructing collections of nodes. The **ASNCache** is a static structure. You cannot add **IASNNodes** to an **ASNCache**; you create it from an **IASNContainer**. An **ASNCache** has the necessary functionality to cache data across process boundaries, and control when it gets copied to and from the simulation.

IASNContainer

The **IASNContainer** holds an arbitrary collection of **IASNNodes**. It can be created, have **IASNNodes** added to it, and be merged with other **IASNContainers**. It can be saved as an XML description of its contents and restored at a later time.

An **IASNContainer** has the restriction that all nodes which it contains must come from the same simulation.

Attributes for nodes within an **IASNContainer** can be queried or set in a single call.

Creating and Modifying an IASNContainer

The **IASNContainer** may be created from a node, in which case it contains only that node:

```
using Aspentech.ASI;  
IASNContainer container = node.CreateContainer();
```

IASNContainer inherits from the .NET interface **ICollection**, so you can add and insert other nodes or containers:

```
using Aspentech.ASI;  
container.Add(node);  
container.Insert(node, 2);  
IASNNode node2 = container.Item[0];  
container.Remove(node2);
```

In addition, you can merge another container into a container using the **Add** method. Remember that all nodes in any container must come from the same simulation.

```
using Aspentech.ASI;  
container.Add(container2);
```

Saving and Restoring an IASNContainer

An IASNContainer can be backed up as an XML string. This can be used later (possibly in another session) to restore the container. You must open the file and obtain any IASNNode to restore the container. This is because the container XML string does not contain any file information.

Save

```
using Aspentech.ASI;
string contents = container.Backup();
```

Restore

```
using Aspentech.ASI;
ASNBroker broker = new ASNBroker();
ASNDocument apDoc = broker.CreateObjectFromFile("c:\\test\\pfdtut.bkp");
IASNNode node = apDoc.GetService(ServiceType.Node);
IASNContainer newcontainer = node.RestoreContainer(contents);
```

IASNContainer Attributes

Attributes of nodes in a container can be obtained through the **GetAttribute** method. **GetAttribute** fills in an existing array of objects (sized to the number of nodes in the container) with attributes:

```
using Aspentech.ASI;
int count = container.Count;
object array = Array.CreateInstance(typeof(object), count);
container.GetAttribute("value", array);
ASNAttribute attrOfFirstNode = array[0];
```

Attributes can be set by passing an array of values to the **SetAttribute** method:

```
using Aspentech.ASI;
int count = 2;
object array = Array.CreateInstance(typeof(double), count);
array[0] = 1.5;
array[1] = 2.0;
container.SetAttribute("value", count, array);
```

IASNContainer Notifications

IASNContainers raise a **DataChanged** event when any contained node is modified or deleted. Attach to the event in the same way as an IASNNode:

```
using Aspentech.ASI;
class containerholder
{
    IASNContainer myContainer;

    public containerholder(IASNContainer container)
    {
        // Upon creation of this class instance,
        // attach notification request
        myContainer = container;
        myContainer.DataChanged += new
            EventHandler(contain_DataChanged);
    }

    protected override void Dispose( bool disposing)
    {
        if (disposing)
            myContainer.DataChanged -=
                new EventHandler(contain_DataChanged);
        base.Dispose(disposing);
    }

    private void contain_DataChanged(object sender, EventArgs args)
    {
        // This code executed when container fires
        // DataChanged event.
        // myContainer and sender will be the same object
    }
}
```

ASNCache

An ASNCache is a local copy of the data from a simulation. It is created from an IASNContainer, restored from a saved cache, or created from the contents of the clipboard. As such it is backed by an IASNContainer, either explicitly given to it or created during creation of the cache. The backing IASNContainer is what maintains the link to the simulation data. The ASNCache is simply a local table of attribute values.

Creation and Initialization

The ASNCache can be created from an IASNContainer, from the contents of the clipboard, from the children of a node, or from the backup of a cache from an earlier session.

In each case, the cache must be initialized to establish the connection to the simulation and to get and/or store the latest data. Before initialization, you can determine which attributes are stored in the cache by calling **SetFilter**. If

SetFilter is not called for a new cache, then all attributes that can be found for all nodes in the cache are stored.

Example 1

```
using Aspentech.ASI;

ASNCache cache = new ASNCache(container);
cache.AccumulateChanges = true;
string[] attrs = {"id", "value", "uom"};
cache.SetFilter(3, attrs);
cache.Initialize();
```

Example 2

```
using Aspentech.ASI;

ASNCache cache = node.CreateCacheOfChildren();
string[] attrs = {"id", "value", "uom", "display_name"};
cache.SetFilter(4, attrs);
cache.Initialize();
```

Example 3

```
using Aspentech.ASI;

ASNDocument doc;

ASNCache cache = node.CreateCacheFromClipboard(out doc);
string[] attrs = {"id", "value", "uom", "display_name"};
cache.SetFilter(4, attrs);
cache.Initialize();

string backupstring = cache.Backup();
using Aspentech.ASI;

IASNNode node = doc.GetService(ServiceType.Node);

ASNCache cache = new ASNCache(backupstring);
cache.Connect(topnode, UpdateMode.DontSendValues);
cache.Initialize();
```

Synchronization

The frequency of synchronization between the cache and simulation is determined by the **AutoUpdate** property. If true, the cache synchronizes with the simulation every time it is notified by the simulation that data has changed, and whenever data is stored in the cache by the caller. That is, if AutoUpdate is true, every time you call SetAttribute or SetAttributeValue the data will be copied from the cache to the simulation. Everytime any node in the backing IASNContainer is notified by the simulation that data has changed, the data will be copied from the simulation to the cache.

If AutoUpdate is false, the cache is only synchronized when the Update method is called. The data that is copied, and the direction of copy, is determined by the **UpdateMode** argument to the Update method, and the **AccumulateChanges** property.

AccumulateChanges property:

Determines what data is considered dirty.

False Data which the caller has set since the last update only.

True Data which the caller has ever modified.

Update Mode:

Default Send dirty data to simulation. Read all data back.

ForceValues Send all data to simulation.

DontSendValues Do not send any data to simulation. Read all data back.

ReapplyValues Send all dirty data to simulation. Do not read data back.

Accessing and Setting Data

Data can be read from the cache through the **GetAttributeValue** and **GetAttribute** calls. **GetAttributeValue** gets a single attribute value for a single node in the cache:

```
using Aspentech.ASI;
ASNAttribute valattr = cache.GetAttributeValue(0, "value");
ASNAttribute unitsattr = cache.GetAttributeValue(0, "uom");
double val = valattr.Value;
string units = unistattr.Value;
```

Use **GetAttribute** to get an array of attributes for all nodes in the cache:

```
using Aspentech.ASI;
ASNAttribute[] values;
values = cache.GetAttribute(2, "value");
double firstval = values[0];
string secondval = values[1];
```

When setting a single attribute value, use **SetAttributeValue**:

```
using Aspentech.ASI;
bool success = cache.SetAttributeValue(0, "value", 1.2);
bool success = cache.SetAttributeValue(0, "uom", "kg");
```

When setting an attribute for all nodes in a cache, use **SetAttribute**. Use **GetAttribute** to get the initial array of values, and modify the values of interest. Only values which you have changed get marked dirty.

```
using Aspentech.ASI;

ASNAttribute[] values;
values = cache.GetAttribute(4, "value");

values[0] = 1.2;
values[4] = "new string value";

cache.SetAttribute("value", 4, values);
```

Solving

ASI currently supports the sequential modular solve, through the **ASNSMSolve** object and the associated **IASNMessageService** interface. The solve interface controls the solution process and the message service filters the message events which occur. Obtain the ASNSMSolve object from the **ASNDocument.GetService** method, and the IASNMessageSupport interface from the solver:

```
using Aspentech.ASI;

ASNSMSolve solver =
    apDoc.GetService(ServiceType.SequentialModularSolve);
```

Messages

The solver sends back messages which can be saved or displayed to the user. The messages can arrive as events or as a returned value from the solve method (if synchronous). These messages are chained in **ASNMessageEventArgs** objects.

Messages have a severity, system, and message number, which can be used for filtering the messages. The severity is one of the **MessageSeverity** enum values. The values for system and number are assigned by the specific simulator.

Sample function to display messages whose severity is less than a given value (assuming that a general purpose Display function is available):

```
using Aspentech.ASI;

public void DisplayMsgs(ASNMessageEventArgs args, int severity)
{
    ASNMessageEventArgs localargs = args;
    while (localargs)
    {
        // Note that, for historical reasons, the enum values
        // for MessageSeverity are in reverse order.
        // Most severe is -1 and least severe is 5

        if (localargs.Severity() < severity)
        {
            // assume a Display function exists
            Display(localargs.HeaderText(), localargs.Text());
            localargs = localargs.Next();
        }
    }
}
```

Solve Methods

A simulation can be queried to see if it can be solved. It will return a true if it can be. Any messages which it generates (such as reasons why it cannot solve) will be returned as an **ASNMessageEventArgs** chain.

```
using Aspentech.ASI;

ASNMessageEventArgs args;

if (solver.CanSolve(out args) == false)
{
    DisplayMsgs(args, 1);
}
else
{
    bool solveStarted = solver.SynchronousSolve(out args);
}
```

The synchronous solve method waits until the solve is complete before returning, and returns a chain of messages. The return code simply indicates that the solve process successfully started. It should always be true if **CanSolve** returned true.

ASI does not prescribe that the solve tells you whether it solved or not, since it may solve with errors, warnings, etc. The message chain can be searched for messages of error severity.

An asynchronous solve returns immediately. It will return false if CanSolve returned false or if asynchronous solve is not supported. It raises Message events, as well as a success or failure event. In general, it will raise the

success event unless a disastrous failure occurs. The following code will do an asynchronous solve and display all messages as they occur:

```
using Aspentech.ASI;

public void ASolve()
{
    ASNMessageEventArgs args;
    if (solver.CanSolve(out args) == false)
    {
        DisplayMsgs(args, 1);
    }
    else
    {
        solver.Message += new
            ASNMessageEventHandler(DisplayMessage);
        solver.Solved += new ASNMessageEventHandler(SolveComplete);
        solver.Failed += new ASNMessageEventHandler(SolveComplete);

        bool solveStarted = solver.AsynchronousSolve(out args);
    }
}

private void DisplayMessage(object solvobj, ASNMessageEventArgs args)
{
    DisplayMsgs(args, -1);
}

private void SolveComplete(object solvobj, ASNMessageEventArgs args)
{
    solver.Message -= new ASNMessageEventHandler(DisplayMessage);
    solver.Solved -= new ASNMessageEventHandler(SolveComplete);
    solver.Failed -= new ASNMessageEventHandler(SolveComplete);
    DisplayMsgs(args, 1000);
}
```

Message Filtering

An asynchronous solve can put in a filter to avoid being overwhelmed with messages, using the **IASNMessageSupport** interface. A synchronous solve can also filter the messages returned from the `SynchronousSolve` method. The message support is a cumulative list of filter commands. When a message is generated it is compared against each filter command which has been made on the message support interface and, if it passes all, it is raised as an event, or returned.

```
using Aspentech.ASI;
ASNSMSolve solver =
    apDoc.GetService(ServiceType.SequentialModularSolve);
IASNMessageSupport messagesupport = solver.GetMessageSupport();
// Suppress all messages which are less severe than error
messagesupport.FilterMessageSeverity(MessageSeverity.Error,
    Filter.SuppressLower,
    FilterDestination.AllDestinations);
// Suppress all messages from subsystem "2" for asynchronous solve
messagesupport.FilterMessageSystem(2,
    Filter.SuppressOnly,
    FilterDestination.EventDestination);
// Suppress all messages from subsystem "3" for synchronous solve
messagesupport.FilterMessageSystem(3,
    Filter.SuppressOnly,
    FilterDestination.ReturnDestination);
```

2 Function Reference

This section contains a reference guide for all methods, properties, enums, etc. for the ASI .NET system.

Services

ASI nodes supply various services, such as creating children, connecting, running, etc. These are supported through interfaces. Rather than use the COM style **QueryInterface** method to expose these interfaces, each interface has a **GetService** method. This allows more flexibility than QueryInterface (similar services can use the same interface, such as inlet ports and outlet ports), and avoids the execution delay of creating an exception in VB when inquiring for a service that does not exist. The possible services are enumerated:

enum ServiceType

Values:

- Broker
- SequentialModularSolve
- Node
- MessageSupport
- ApplicationUnknown

Messages and Errors

Simulators can issue diagnostic messages as well as raise error conditions (exceptions). This might happen during a synchronous call (such as store value, connect port, CanSolve, or solve) or an asynchronous operation (solve).

Note that failure to find a child node is not an error. Raising an exception in such a situation would slow searches down considerably. A simulator should only raise an exception if it cannot continue to perform the function it was asked to perform. For example, simply encountering a zero flow in a block calculation should not raise an exception, since simulation continues.

Messages are returned to the caller either as events, as returned message arrays, or as both. They have a diagnostic level tag, a subsystem tag, and a

tag specific to a given message. These tags can be used to suppress messages in a flexible manner.

enum Filter

Values:

- SuppressLower (applies to FilterMessageSeverity only)
- SuppressOnly
- SuppressExcept

enum FilterDestination

Values:

- EventDestination filters events raised
- ReturnDestination filters array returned from function
- AllDestinations filters both

enum MessageSeverity

Values:

- SystemError
- TerminalError
- SevereError
- Error
- Warning
- Information
- Diagnostic

Interface IASNMessageSupport

To create:

- Obtain from IASNNode:GetService(ServiceType type)
- Obtain from IASNSolve:GetService(ServiceType type)

Methods:

- ClearFilter(FilterDestination destination);
- SuppressAll(FilterDestination destination);
- FilterMessageSeverity(MessageSeverity level, Filter action, FilterDestination destination);
- FilterMessageSystem(int system, Filter action, FilterDestination destination);
- FilterMessageNumber(int number, Filter action, FilterDestination destination);

Class ASNMessageEventArgs : System.EventArgs

get Methods:

- int Severity
- int System
- int Number
- string HeaderText Line of text containing above three values
- string Text Text of the error
- ASNMessageEventArgs Next Gets next error in array of errors

Creating Objects

Objects are created through the ASI broker interface. A top level ASI broker object can be created that is a factory for any ASI object that can exist without a containing framework. Such objects would include full simulations, standalone models, tools that work with simulations (such as case comparators), etc.

Objects that can only be created by a supporting framework, and can only exist within that framework, are created by the ASI broker interface that is exposed as a service by some node in the framework's hierarchy. For example, the flowsheet level node in a closed simulator would expose an ASI broker interface, which would be used to create unit operation blocks.

Object ASNBroker

Create:

- ASNBroker broker = new ASNBroker();
- ASNDocument doc = broker.CreateDocumentFromFile(string file);
- IASNNode node = doc.GetService(ServiceType type);
- Obtain from ASNDocument.GetService(ServiceType type)
- Obtain from IASNNode.GetService(ServiceType type)
- Obtain from IASNSolve.GetService(ServiceType type)

Methods:

- ASNDocument CreateDocumentFromFile(string file);
 - Creates a document object for the file.
- ASNDocument CreateDocumentFromFileVersion(string file, string version);
 - Creates a document object for the file using the specified version of the application.
- ASNCache CreateCacheFromClipboard(out ASNDocument doc);
 - Creates an ASNCache object from the contents of the clipboard, and returns the ASNDocument for the clipboard contents.
- IASNNode CreateObjectFromName(string name);
- IASNNode CreateObjectFromType(string type);

- IASNNode CreateObjectFromXml(string xmlDesc);
 - Creates an object of the given type. If file is specified, uses file to initialize object. If xmlDesc is specified, used by application as needed. Returns top node in the created object's hierarchy of nodes. Returned node can provide ASNSolve if appropriate.

Object ASNDocument

Create:

- ASNBroker broker = new ASNBroker();
- ASNDocument doc = broker.CreateDocumentFromFile(string file);

Methods:

- void Close();
Closes the document. All objects from the document are invalid following the Close().
- object GetService(ServiceType type);
Returns object for ServiceTypes ApplicationUnknown, Broker, Node, and SequentialModularSolve
- bool Save();
Saves any changes to the document.
- bool SaveAs(string filename);
Saves any changes to the document in the file specified.

Properties (Get):

- string FileName
The full path of the open document.
- string AppName
The application name of the open document.
- string Version
The application defined version string for the open Document.
- string Status
The application defined status string for the state of the simulation.

Accessing an Object's Data

An object's data values and attributes are accessed and set through the ASxNode and ASxContainer interfaces. These are a refinement of the ASiv1 interfaces.

Interface IASNNode

: System.Collections.ICollection, System.IComparable

Create:

- Obtain from ASNBroker object.
- Obtain from child or ancestor IASNNode.
- Obtain from ASNNodeEnumerator.
- Obtain from ASNContainerEnumerator

Properties:

// ICollection

- int count { get; }

// IASNNode

- IDictionary Attributes { get; }
Returns an IDictionary of ASNAttribute objects.

Methods:

// IEnumerable

- IEnumerator GetEnumerator();

// ICollection

- void CopyTo(Array array, int index);
Returns an Array of child IASNNodes.

//IASNNode

- object GetService(ServiceType type);
- IASNNode GetChild(object path_or_index);
- IASNNode GetParent();
- string GetXML(int numatts, Array attrs);
- int GetXMLSize(int numatts, Array attrs);
- bool IsLeaf()
- bool IsValueNULL()
- IASNContainer CreateContainer()
- IASNContainer CreateClipboardContainer()
- IASNContainer RestoreContainer(string xml)
- IASNNodeLink GetLink()
Creates ASNNodeLink which will update target when this node's value changes.
- ASCache CreateCacheOfChildren()
- string GetPath()

Events:

1: DataChanged

2: Deleted

enum AttributeType

Values:

- Empty
- Integer
- Real
- String

Object ASNAttribute

Create:

- Obtain from Attributes IDictionary object

Properties:

- AttributeType Type {get;}
- object Value {get; set}
- bool IsCached {get;}

Casts:

ASNAttributes can be implicitly cast to get the value for the correct attribute type

Interface IASNContainer

: System.Collections. IList

Create:

- Obtain from IASNNode:CreateContainer
- Obtain from IASNNode:RestoreContainer
- Obtain from IASNNode:CreateClipboardContainer

Properties:

// ICollection

- int Count { get; }

// IList

- bool IsFixedSize { get; }
- bool IsReadOnly { get; }
- IASNNode Item[int index] { get; }

Methods:

// IEnumerable

- IEnumerator GetEnumerator();
Returns an enumerator of the contained IASNNodes

// ICollection

- void CopyTo(Array array, int index);

```
// IList
• int Add(object value);
  Add IASNNode or IASNContainer

• void Clear();
• bool Contains(object value);
• int IndexOf(object value);
• void Insert(int index, object value);
• void Remove(object value);
• void RemoveAt(int index);

//IASNContainer
• Array GetAttribute(string attrname);
• String SetAttribute(string attrname, int length, Array values);
• IASNContainerLink GetLink();
• string Backup();
  Returns XML string suitable for IASNNode:RestoreContainer

Events:
1: DataChanged
2: Deleted
```

object ASNCache : IList, IDisposable

When IList methods of the ASNCache are used, the ASNAttribute objects get and set values in the Cache, as opposed to ASNAttribute objects obtained from IASNNode objects that get and set values directly in the ASNDocument.

Create:

- new ASNCache(IASNContainer container);
- new ASNCache(string strXml);
- Obtain from IASNNode.CreateCacheOfChildren
- Obtain from ASNBroker. CreateCacheFromClipboard

Internal Enums:

- UpdateMode
 - Default - if dirty then send values to simulation
 - ForceValues - send values to simulation regardless of dirty flag
 - DontSendValues - do not send values to simulation regardless of dirty flag
 - ReapplyValues – send values to simulation from the accumulated change list.

Properties:

```
// ICollection
• int Count { get; }

// IList
• bool IsFixedSize { get; }
```

- `bool IsReadOnly { get; }`
- `IDictionary Item[int index] { get; }`
Returns an `IDictionary` of `ASNAttributes` for the given index. The `IDictionary` object is similar to value returned by the `IASNNode.Attributes` property.

// `ASNCache`

- `Array Attributes {get;}`
Returns a string array listing all possible attributes for the contained nodes.
- `bool AutoUpdate { get; set; }`
- `bool AccumulateChanges { get; set }`
If true, the cache maintains a list of all values modified by calls to `SetAttribute` and `SetAttributeValue`.

Methods:

// `IEnumerable`

- `IEnumerator GetEnumerator();`
Returns an `IEnumerator`, which enumerates through the index values, returning an `IDictionary` of `ASNAttributes` for the given index.

// `ICollection`

- `void CopyTo(Array array, int index);`
Returns an array of `IDictionary` objects which contain `ASNAttributes` for the given index.

// `IList`

All `IList` methods will take an `IASNNode` or an `IDictionary` of `ASNAttributes` (from the `IDictionary`, `IEnumerable`, or `ICollection` methods of the `ASNCache`) as the object.

- `bool Contains(object value);`
- `int IndexOf(object value);`

// `ASNCache`

- `string[] AttributeList();`
Returns a list of the active attributes, based on the filter.
- `void SetFilter(int numatts, string[] attlist);`
- `void Initialize();`
- `void SetUpdateNode(IASNNode node);`
- `void Update();`
Update cache from container, update values from application
- `void Update(UpdateMode mode);`
Update values based on mode
- `void Connect(IASNNode topNode);`
- `void Connect(IASNNode topNode, UpdateMode mode);`
- `void Disconnect();`

- `TypeCode GetAttributeType(int index, string attrname);`
- `object GetAttributeValue(int index, string attrname);`
- `Array GetAttribute(string attrname);`
- `bool SetAttributeValue(int index, string attrname, object value);`
- `string SetAttribute(string attrname, int length, Array values);`
- `string Backup();`
Backup XML.
- `IASNNode TopNode();`
- `void RemoveFromChangeList(int index, string attrname);`
Removes the change for node index and attribute from the accumulated change list.

Events:

`//ASNCache`

- `DataChanged`

Interface IASNNodeLink

Create:

- Obtain from `IASNNode:GetLink`

Methods:

- `void AddTarget(IASNNode target);`
- `void Delete();`
- `void RemoveTarget();`
- `void Trigger();`

Interface IASNContainerLink

Create:

- Obtain from `IASNContainer:GetLink`

Methods:

- `void AddTarget(IASNContainer target);`
- `void Delete();`
- `void RemoveTarget();`
- `void Trigger();`

Solving Objects

An object may exhibit one or more types of solving behavior. For example, a unit operation may solve in sequential modular mode (read input streams, compute output streams), equation oriented (use variables' spec attribute to determine fixed and free variables), reverse flow (read output streams, compute input streams), etc. For each behavior, there is a solve interface.

The number and complexity of these interfaces is expected to grow during development.

Objects may support both synchronous and asynchronous solving behavior. For asynchronous operations, a connection point (COM) or complete event is supported.

Solver objects may support attributes that describe features and the state of the solver object.

Interface IASNSMSolve

Create:

- Obtain from `IASNNode::GetService(ServiceType type)`
- Obtain from `IASNSolve::GetService(ServiceType type)`

Properties:

- `IDictionary Attributes { get; }`
Returns an `IDictionary` of `ASNAttribute` objects

Methods:

- `void Initialize();`
Initializes the current solution to its initial state. May purge the current results from the problem.
- `bool CanSolve(out ASNMessageEventArgs msgs);`
- `bool SynchronousSolve(out ASNMessageEventArgs msgs);`
- `bool AsynchronousSolve();`
Starts an asynchronous solve task. Returns false if the solve task cannot be started. A `SolveEvent` is fired when the solve completes or is paused.
- `void Pause();`
- `void Stop();`
- `void Resume();`
- `void Restart();`
Restart the solution from its initial state.
- `IASNMessageSupport GetMessageSupport();`

Events:

- 1: `ASNMessageEvent Solved;`
- 2: `ASNMessageEvent Failed;`
- 3: `ASNMessageEvent Message;`

Appendix A Standard Attributes

Clients of ASI can make much greater use of ASI if there is uniform way to find information. To address this problem at the data model level, we are creating the simulation-based CCDM data model. This allows the user to traverse the data hierarchy in a known manner.

However, if ASI is supported by a wide range of applications, the CCDM data model will not apply, and clients will traverse the data in the supporting application's data model. In some circumstances, it will also be useful to traverse an application's native data model even if the application does support CCDM.

If general-purpose tools, such as browsers and variable managers, are to be written to work with ASI applications, it is necessary to achieve standardization beyond the CCDM data model level. As much as possible, the names and uses of attributes at the node level within the hierarchy should be standardized. If applicable information is supported by the application, it should use the following node attributes:

value

Object's value if it is a leaf. Should be real, integer, string, or enum.

dimension

Object's dimension (e.g. temperature).

uom

Displayed units of measure. Setting a uom attribute does not change the unit in the underlying application. It changes the unit that values are displayed in by the ASI interfaces. Best if CDM units.

spec_value

Value which user has specified. May be different from value.

spec_uom

Units that spec-value are displayed in.

can_enter

Value may be entered.

option_list

List of options for value. For each option, value and optional description, separated by tab. Carriage return separates options. Example:

```
option1 \t description \n
option2 \n
option3
```

uom_options

List of options for **uom**.

local_name

Name to uniquely distinguish the object in the context of its container

type

Object's type. If leaf, should be "Real", "Integer", "String", or "EnumXXX".

container

Display_name of object that contains **this**. For simulation, should be the unit operation.

container_type

Type of object that contains **this**.

display_name

Shortest name to distinguish object from its siblings in context of immediate parent. Examples:

- Stream.Temperature.MIXED
Use MIXED
- Flowsheet.B1
Use "UnitOperation B1"
- Flowsheet.S1
Use "Stream S1"
- Flowsheet.BLOCKS.B1
Use B1

isleaf

Static. 1 if leaf, 0 if not.

has_children

Dynamic. 0 if leaf, 0 if not leaf but currently has no children.

path

Path that can be presented to top node GetChild method, to get to this node.

upper_limit

Lower limit in current spec-uom if spec-uom is entered, else in uom

lower_limit

Lower limit in current spec-uom if spec-uom is entered, else in uom

description

Description.

Appendix B Path Specification (BNF Description)

The path is used to reference children of a given node. It may, depending on the context, retrieve a child, or it may get the value of an attribute of the given child (In a conditional, for example.)

It is evaluated left to right, following the application iterator. If a match is found, evaluation continues. If a value is required, the trailing [<attrname>](#) is used to determine which attribute is queried. If the [<attrname>](#) is not present, the value attribute is assumed.

<code><path> ::=</code>	<code>[<pathlist>] ["(" <attrname> ")"]</code>	
<code><pathlist> ::=</code>	<code>[<name>] ["#" <count>] [<test>]</code> <code>["." <pathlist>]</code>	
<code><name> ::=</code>	<code><sval> "*"</code>	name of element at this level. * matches any element
<code><count> ::=</code>	<code><ival></code>	0 based integer specifying element number (if <code><name></code> is specified, this goes to nth element of that name.)
<code><test> ::=</code>	<code>["!"] ["(" <path> ["=" <sval>] ")"]</code> <code>[["&" " "] <test>]</code>	
<code><attrname> ::=</code>	<code><sval></code>	name of attribute (defaults to "value")
<code><sval> ::=</code>	<code><word> "%" <word></code>	
<code><ival> ::=</code>	<code>integer "%" <word></code>	
<code><word> ::=</code>	<code>single word with no punctuation </code> <code>"" any string ""</code>	

Path Examples:

- child
- child#4
- child.child
- child[(value)=Fred]. child[(name)=%gcname](age)
Finds the age of your son Fred's child who's name is stored in the variable gcname.

General Information

Copyright

Version Number: 2004.1

April 2005

Copyright © 2004 – 2005 Aspen Technology, Inc. All rights reserved.

Aspen Accounting.21™, Aspen ACOL™, Aspen Adsim®, Aspen Advisor™, Aspen AeroTran®, Aspen Alarm & Event™, Aspen APLE™, Aspen Apollo™, Aspen AssetBuilder Optimizer™, Aspen AssetBuilder Planner™, Aspen AssetBuilder Site Optimizer™, Aspen AssetBuilder™, Aspen Batch Plus®, Aspen Batch.21™, Aspen Batch.21™ CBT, Aspen BatchCAD™, Aspen BatchSep™, Aspen Blend Model Library™, Aspen Blend™, Aspen Calc™ CBT, Aspen Capable-to-Promise®, Aspen CatRef®, Aspen Chromatography®, Aspen Cim-IO Interfaces™, Aspen Cim-IO Monitor™, Aspen Cim-IO™ for @AGlance, Aspen Cim-IO™ for ABB 1190, Aspen Cim-IO™ for Bailey SemAPI, Aspen Cim-IO™ for DDE, Aspen Cim-IO™ for Eurotherm Gauge via CDP, Aspen Cim-IO™ for Fisher-Rosemount Chip, Aspen Cim-IO™ for Fisher-Rosemount RNI, Aspen Cim-IO™ for Foxboro FOXAPI, Aspen Cim-IO™ for G2, Aspen Cim-IO™ for GE FANUC via HCT, Aspen Cim-IO™ for Hitachi Ex Series, Aspen Cim-IO™ for Honeywell TDC 3000 via HTL/access, Aspen Cim-IO™ for Intellution Fix, Aspen Cim-IO™ for Measurex MCN, Aspen Cim-IO™ for Measurex ODX, Aspen Cim-IO™ for Moore Apacs via Nim (RNI), Aspen Cim-IO™ for PI, Aspen Cim-IO™ for RSLinx, Aspen Cim-IO™ for SetCim/InfoPlus-X/InfoPlus.21, Aspen Cim-IO™ for Toshiba Tosdic, Aspen Cim-IO™ for ULMA 3D, Aspen Cim-IO™ for Westinghouse, Aspen Cim-IO™ for WonderWare InTouch™, Aspen Cim-IO™ for Yokogawa ACG10S, Aspen Cim-IO™ for Yokogawa EW3, Aspen Collaborative Forecasting™, Aspen Compliance.21™, Aspen COMThermo TRC Database™, Aspen COMThermo®, Aspen Crude Trading & Marketing™, Aspen Custom Modeler®, Aspen Decision Analyzer™, Aspen Demand Manager™, Aspen DISTIL™, Aspen Distribution Scheduler™, Aspen DMCplus®, Aspen DMCplus® CBT, Aspen DMCplus® Composite, Aspen DPO™, Aspen Dynamics®, Aspen eBRS™, Aspen ERP Connect®, Aspen FCC®, Aspen FIHR™, Aspen FLARENET™, Aspen Fleet Operations Management™, Aspen FRAN™, Aspen Fuel Gas Optimizer™, Aspen Grade-IT™, Aspen Harwell Subroutine Library™, Aspen Hetran®, Aspen HTFS Research Network™, Aspen HX-Net Operations™, Aspen HX-Net®, Aspen Hydrocracker®, Aspen Hydrotreater™, Aspen HYSYS Amines™, Aspen HYSYS Crude™, Aspen HYSYS Data Rec™, Aspen HYSYS Dynamics™, Aspen HYSYS Johnson Matthey Reactor Models™, Aspen HYSYS OLGAS 3-Phase™, Aspen HYSYS OLGAS™, Aspen HYSYS OLI Interface™, Aspen HYSYS Optimizer™, Aspen HYSYS Tacite™, Aspen HYSYS Upstream Dynamics™, Aspen HYSYS Upstream™, Aspen HYSYS®, Aspen Icarus Process Evaluator®, Aspen Icarus Project Manager®, Aspen InfoPlus.21®, Aspen Inventory Management & Operations Scheduling™, Aspen Inventory Planner™, Aspen IQmodel Powertools™, Aspen IQ™, Aspen Kbase®, Aspen Lab.21, Aspen MBO™, Aspen MPIMS™, Aspen Multivariate Server™, Aspen MUSE™, Aspen OnLine®, Aspen Operations Manager - Event Management™, Aspen Operations Manager - Integration Infrastructure™, Aspen Operations Manager - Integration Infrastructure™ Aspen Advisor, Aspen Operations Manager - Integration Infrastructure™ Aspen Orion, Aspen Operations Manager - Integration Infrastructure™ Aspen PIMS, Aspen Operations Manager - Integration Infrastructure™ Aspen Utilities, Aspen Operations Manager - Integration Infrastructure™ Base, Aspen Operations Manager - Integration Infrastructure™ COM, Aspen Operations Manager - Integration Infrastructure™ Files, Aspen Operations Manager - Integration Infrastructure™ IP.21, Aspen Operations Manager - Integration Infrastructure™ IP.21/SAP-PPPI, Aspen Operations Manager - Integration Infrastructure™ OPC, Aspen Operations Manager - Integration Infrastructure™ Relational Databases, Aspen Operations Manager - Integration Infrastructure™ SAP R3, Aspen Operations Manager - Integration Infrastructure™ System Monitoring, Aspen Operations Manager - Performance Scorecarding™, Aspen Operations Manager - Role Based Visualization™ MS SharePoint, Aspen Operations Manager - Role Based Visualization™ TIBCO, Aspen Operations Tracking™, Aspen Order Credit Management™, Aspen Orion Planning™, Aspen Orion XT™, Aspen PEP Process Library™, Aspen PIMS Distributed Processing™, Aspen PIMS Enterprise Edition™, Aspen PIMS Global Optimization™, Aspen PIMS Mixed Integer Programming™, Aspen PIMS Simulator Interface™, Aspen PIMS Solution Ranging™, Aspen PIMS Submodel Calculator™, Aspen PIMS XNLP Optimizer™, Aspen PIMS™, Aspen PIPE™, Aspen Plant Planner & Scheduler™, Aspen Plant Scheduler Lite™, Aspen Plant Scheduler™, Aspen Polymers Plus®, Aspen Plus OLI Interface™, Aspen Plus Optimizer™, Aspen Plus®, Aspen Plus® CBT, Aspen PPIMS™, Aspen Process Explorer™ CBT, Aspen Process Manual™ Applied Rheology, Aspen Process Manual™ Bulk Solids Handling, Aspen Process Manual™ Crystallization, Aspen Process Manual™ Drying, Aspen Process Manual™ Gas Cleaning, Aspen Process Manual™ Internet Mode, Aspen Process Manual™ Intranet Mode, Aspen Process Manual™ Mini-Manuals, Aspen Process Manual™ Slurry Handling, Aspen Process Manual™ Solid Liquid Separation, Aspen Process Manual™ Solvent Extraction, Aspen Process Manual™ Waste Water Treatment, Aspen Process Order™, Aspen Process Recipe®, Aspen Process Tools™, Aspen Production Control Web Server™, Aspen ProfES® 2P Wax, Aspen ProfES® Tranflo, Aspen Profile.21™, Aspen Properties®, Aspen Pumper Log™, Aspen Q Server™, Aspen Quality Management™, Aspen RefSYS CatCracker™, Aspen RefSYS™, Aspen Report Writer™, Aspen Retail Automated Stock Replenishment™, Aspen Retail Resource Scheduling Optimization™, Aspen Richardson Rbooks™, Aspen Richardson WinRace Database™, Aspen RTO Watch™, Aspen SCM™, Aspen SmartStep Advanced™, Aspen SmartStep™, Aspen Specialty Products Automated Stock Replenishment™, Aspen Specialty Products Resource Scheduling Optimization™, Aspen Split™, Aspen SULSIM®, Aspen Supply Chain Analytics™, Aspen Supply Chain Connect™, Aspen Supply Planner™, Aspen Tank Management™, Aspen TASC™, Aspen Teams®, Aspen TICP™, Aspen Transition Manager™, Aspen Utilities™, Aspen

Voice Fulfillment Management™, Aspen Watch™, Aspen Water™, Aspen Web Fulfillment Management™, Aspen XPIMS™, Aspen Zyqad Development™, Aspen Zyqad™, SLM™, SLM Commute™, SLM Config Wizard™, the Aspen leaf logo, and Plantelligence are trademarks or registered trademarks of Aspen Technology, Inc., Cambridge, MA.

All other brand and product names are trademarks or registered trademarks of their respective companies.

This document is intended as a guide to using AspenTech's software. This documentation contains AspenTech proprietary and confidential information and may not be disclosed, used, or copied without the prior consent of AspenTech or as set forth in the applicable license.

Although AspenTech has tested the software and reviewed the documentation, the sole warranty for the software may be found in the applicable license agreement between AspenTech and the user. ASPENTECH MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO THIS DOCUMENTATION, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Corporate

Aspen Technology, Inc.
Ten Canal Park
Cambridge, MA 02141-2201
USA
Phone: (1) (617) 949-1000
Toll Free: (1) (888) 996-7001
Fax: (1) (617) 949-1030
URL: <http://www.aspentech.com/>

Technical Support

Online Technical Support Center

AspenTech customers with a valid license and software maintenance agreement can register to access the Online Technical Support Center at:

<http://support.aspentech.com>

You use the Online Technical Support Center to:

- Access current product documentation.
- Search for technical tips, solutions, and frequently asked questions (FAQs).
- Search for and download application examples.
- Search for and download service packs and product updates.
- Submit and track technical issues.
- Search for and review known limitations.
- Send suggestions.

Registered users can also subscribe to our Technical Support e-Bulletins. These e-Bulletins proactively alert you to important technical support information such as:

- Technical advisories.
- Product updates.
- Service Pack announcements.
- Product release announcements.

Phone and E-mail

Customer support is also available by phone, fax, and e-mail for customers who have a current support contract for their product(s). Toll-free charges are listed where available; otherwise local and international rates apply.

For the most up-to-date phone listings, please see the Online Technical Support Center at:

<http://support.aspentech.com>

Support Centers	Operating Hours
North America	8:00 – 20:00 Eastern time
South America	9:00 – 17:00 Local time
Europe	8:30 – 18:00 Central European time
Asia and Pacific Region	9:00 – 17:30 Local time