

# The Active Tables application Manual

## Version 1.2-pre-alpha

\*The Java CoG Kit Team  
Argonne National Laboratory  
Mathematics and Computer Science Division  
9700 S. Cass Ave  
Argonne, IL 60439

\* Corresponding Editor  
(630) 252 0472  
[gregor@mcs.anl.gov](mailto:gregor@mcs.anl.gov)

Location of Manual:

<http://www.globus.org/cog/projects/cmcs/current/manual->

Be kind to your environment and  
*do not print*  
this frequently changing manual.



(c) Argonne National Laboratory. All rights reserved.

February 24, 2004

---

# Contents

---

<b>1 License</b>	<b>4</b>
1.1 General Comments . . . . .	4
1.2 Globus Toolkit Public License (GPL) . . . . .	4
1.3 Other Licences . . . . .	6
1.3.1 Active Tables Web Service . . . . .	6
1.3.2 Active Tables Client . . . . .	6
<b>2 Introduction</b>	<b>8</b>
2.1 Mission . . . . .	8
2.2 ANL's Role . . . . .	8
2.3 Active Tables Framework . . . . .	8
<b>3 Architecture and Design</b>	<b>10</b>
3.1 Introduction . . . . .	10
3.2 Background . . . . .	11
3.2.1 Support for Dynamic Users . . . . .	13
3.3 Scenarios of usage . . . . .	13
3.3.1 Administrator User . . . . .	13
3.3.2 Registered User . . . . .	14
3.3.3 Guest User . . . . .	14
3.4 Design of the Active Tables Web Service . . . . .	15
3.4.1 Interfaces . . . . .	15
3.4.2 Implementation . . . . .	19
3.5 Design of the Active Tables Client . . . . .	20
3.6 Design Issues . . . . .	20
3.6.1 Extensibility . . . . .	21
3.6.2 Security . . . . .	21

<b>4</b>	<b>Development</b>	<b>23</b>
4.1	Notes	23
4.1.1	General	23
4.1.2	AJmenu	24
4.1.3	CMCS Portal development	25
4.2	How To's	26
4.2.1	Testing	27
4.3	Contacts	27
<b>5</b>	<b>Installation</b>	<b>29</b>
5.1	Quick Install guide	29
5.2	Prerequisists	31
5.2.1	What does the Web Service need to run?	31
5.2.2	What does the Portlet need to run?	32
5.3	Download	33
5.3.1	CVS Release Tags	33
5.3.2	Downloading the binary distribution	33
5.3.3	Downloading the sources	33
5.4	Building from the sources	33
5.4.1	Building the ActiveTablesQueryExport.jar	33
5.4.2	Building the Active Tables Web Service	34
5.4.3	Compiling the Web (Portlet) Client for the CMCS portal	35
5.4.4	Compiling the Web (Portlet) Client for Jetspeed	36
5.5	Deployment and Administration	36
5.5.1	Issues	36
5.5.2	Public Web Service on Bubbles	36
<b>6</b>	<b>Usage</b>	<b>37</b>
6.1	Active Tables Web (Portlet) Client	37
6.1.1	Active Tables Web (Portlet) Client for the CMCS Portal	37
6.1.2	Active Tables Web (Portlet) Client for Jetspeed	44
6.2	Active Tables Java Swing Client	44

<b>7 Appendix</b>	<b>49</b>
7.1 FAQs	49
7.2 Achievements	50
7.3 TODO	51
7.3.1 XML & Schema	51
7.3.2 Security	55
7.3.3 Other	55
7.4 Change Log	57
7.4.1 Changes from atct-1-1 to atct-1-2-alpha	57
7.4.2 Changes from atct-1-0 to atct-1-0	57
<b>8 References</b>	<b>58</b>

---

# 1 License

---

## 1.1 General Comments

The Active Tables 1.2.alpha is distributed under two licenses. The Active Tables application consists of two main components, one — the Active Tables wrapper application which is distributed under the Globus Toolkit Public License (GTPL) which is listed in Section 1.2, and two — the Qengine kernel, which is currently not distributable as the license is under construction, but it can be used through the publicly available Web Service.

For further details with regards to the Qengine kernel please contact Branko Ruscic (ruscic@anl.gov), Sandra Bittner (bittner@mcs.anl.gov), and Gregor von Laszewski (gregor@mcs.anl.gov).

The Active Tables application distribution also contains a Web (Portlet) Client for the CMCS Portal. The CMCS Portal has its own licensing. Please talk to David Leahy (djleahy@sandia.gov), or any of the other members of the CMCS team. More information is available at:

<http://www.cmcs.org/>

## 1.2 Globus Toolkit Public License (GTPL)

Copyright (c) 1999 University of Chicago and The University of Southern California. All Rights Reserved.

1. The “Software”, below, refers to the Globus Toolkit (in either source-code, or binary form and accompanying documentation) and a “work based on the Software” means a work based on either the Software, on part of the Software, or on any derivative work of the Software under copyright law: that is, a work containing all or a portion of the Software either verbatim or with modifications. Each licensee is addressed as “you” or “Licensee.”
2. The University of Southern California and the University of Chicago as Operator of Argonne National Laboratory are copyright holders in the Software. The copyright holders and their third party licensors hereby grant Licensee a royalty-free nonexclusive license, subject to the limitations stated herein and U.S. Government license rights.
3. A copy or copies of the Software may be given to others, if you meet the following conditions:
  - (a) Copies in source code must include the copyright notice and this license.
  - (b) Copies in binary form must include the copyright notice and this license in the documentation and/or other materials provided with the copy.

4. All advertising materials, journal articles and documentation mentioning features derived from or use of the Software must display the following acknowledgement:

”This product includes software developed by and/or derived from the Globus project (<http://www.globus.org/>).”

In the event that the product being advertised includes an intact Globus distribution (with copyright and license included) then this clause is waived.

5. You are encouraged to package modifications to the Software separately, as patches to the Software.
6. You may make modifications to the Software, however, if you modify a copy or copies of the Software or any portion of it, thus forming a work based on the Software, and give a copy or copies of such work to others, either in source code or binary form, you must meet the following conditions:

- (a) The Software must carry prominent notices stating that you changed specified portions of the Software.

- (b) The Software must display the following acknowledgement:

“This product includes software developed by and/or derived from the Globus Project (<http://www.globus.org/>) to which the U.S. Government retains certain rights.”

7. You may incorporate the Software or a modified version of the Software into a commercial product, if you meet the following conditions:

- (a) The commercial product or accompanying documentation must display the following acknowledgment:

“This product includes software developed by and/or derived from the Globus Project (<http://www.globus.org/>) to which the U.S. Government retains a paid-up, nonexclusive, irrevocable worldwide license to reproduce, prepare derivative works, and perform publicly and display publicly.”

- (b) The user of the commercial product must be given the following notice:

“[Commercial product] was prepared, in part, as an account of work sponsored by an agency of the United States Government. Neither the United States, nor the University of Chicago, nor University of Southern California, nor any contributors to the Globus Project or Globus Toolkit nor any of their employees, makes any warranty express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

IN NO EVENT WILL THE UNITED STATES, THE UNIVERSITY OF CHICAGO OR THE UNIVERSITY OF SOUTHERN CALIFORNIA OR ANY CONTRIBUTORS TO THE GLOBUS PROJECT OR GLOBUS TOOLKIT BE LIABLE FOR ANY DAMAGES, INCLUDING DIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM EXERCISE OF THIS LICENSE AGREEMENT OR THE USE OF THE [COMMERCIAL PRODUCT].”

8. LICENSEE AGREES THAT THE EXPORT OF GOODS AND/OR TECHNICAL DATA FROM THE UNITED STATES MAY REQUIRE SOME FORM OF EXPORT CONTROL LICENSE FROM THE U.S. GOVERNMENT AND THAT FAILURE TO OBTAIN SUCH EXPORT CONTROL LICENSE MAY RESULT IN CRIMINAL LIABILITY UNDER U.S. LAWS.
9. Portions of the Software resulted from work developed under a U.S. Government contract and are subject to the following license: the Government is granted for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable worldwide license in this computer software to reproduce, prepare derivative works, and perform publicly and display publicly.
10. The Software was prepared, in part, as an account of work sponsored by an agency of the United States Government. Neither the United States, nor the University of Chicago, nor The University of Southern California, nor any contributors to the Globus Project or Globus Toolkit, nor any of their employees, makes any warranty express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.
11. IN NO EVENT WILL THE UNITED STATES, THE UNIVERSITY OF CHICAGO OR THE UNIVERSITY OF SOUTHERN CALIFORNIA OR ANY CONTRIBUTORS TO THE GLOBUS PROJECT OR GLOBUS TOOLKIT BE LIABLE FOR ANY DAMAGES, INCLUDING DIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM EXERCISE OF THIS LICENSE AGREEMENT OR THE USE OF THE SOFTWARE.

END OF LICENSE

### 1.3 Other Licences

We distribute a number of other libraries with the Active Tables application. These libraries come with their own licences. We strongly encourage you to inspect these licenses. They can be found in the “lib” directories of the Active Tables distribution.

#### 1.3.1 Active Tables Web Service

The Web Service “lib” directory contains the following licences.

web-service : [castor.LICENSE](#)

web-service : [apache.LICENSE](#)

The Graphviz, graph drawing software, is available under an open source license, and the license is available in the graphviz “bin” directory.

src/graphviz/bin : [graphviz.LICENSE](#)

#### 1.3.2 Active Tables Client

The Client “lib” directory contains the following licences:

portlet-jetspeed : [apache.LICENSE](#)

Active Tables Web (Portlet) Client(s)

The Web Client(s) “dynamic” directory contains the following licences:

src/portlet-jetspeed/dynamic : [ajmenu.LICENSE](#)

---

## 2 Introduction

---

### 2.1 Mission

To develop a pilot Collaboratory for the Multi-scale Chemical Sciences (CMCS) that will bring together leaders in scientific research and technological development across multiple DOE laboratories, other government laboratories and academic institutions to develop an informatics-based approach to synthesizing multi-scale information to create knowledge in the chemical sciences. The CMCS will use advanced collaboration and metadata-based data management technologies to develop an MCS (Multi-scale Chemical Sciences) portal providing community communications mechanisms and data search and annotation capabilities. The portal will also provide capabilities for defining and browsing cross-scale dependencies between data produced at one scale that is used as input for computations at the next. Notification mechanisms will make both researchers and their applications aware of updated values of relevant information such as reaction rates. The CMCS and its MCS portal will provide mechanisms to enhance the coordination of research efforts across related sub-disciplines in the chemical sciences, focusing research at one scale on obtaining or refining values critical in the next, reducing work performed using limited or outdated values, and enhancing the ability of the community to meet the national research challenges of DOE.

### 2.2 ANL's Role

Our principal focus will be on the development of active thermochemical tables (Active Tables) that provide a computational representation of the relationships between molecular-scale data and derived thermochemical properties of molecules. We will implement intelligent decision-making features in the tables via automated linear analysis and will devise interactive interfaces to the active tables that enable tests of "what-if" scenarios. Our secondary responsibilities will include assisting the development of a Multiscale Chemical Science Portal and the deployment of services that allow access to existing reference databases such as those at NIST.

The main CMCS Project page can be found at :

<http://www.cmcs.org>

The rest of this document will focus on a subset of our contributions.

### 2.3 Active Tables Framework

The Active Tables application consists of two main components. One, the Web Service, which provides global access to the Active Tables application through a set of primitives providing for security, user data manipulation, experiment setup, and query/analysis of the experiments designed. Two, the Active Tables clients, that would reside on the client desktop and provide end users with the options to access the Web Service via a graphical user interface (GUI), be it Web (portlet) based or Java Swing based.

The rest of the document is structured as described here. Chapter 3 describes the architecture and design of the above mentioned components. Chapter 4 contains the developer notes. Chapter 5 describes the steps to be taken to download and install, and administer the Active Tables application release. Chapter 6 contains a small tutorial describing general use cases for the Active Tables application. Chapter 7 contains the Appendices, with the FAQs, Achievement list, TODO list, and the Changelog. Chapter 8 contains the Bibliography.

---

## 3 Architecture and Design

---

This chapter provides information about the architecture and design of the Active Tables application, its various components, their interaction mechanisms, and their working, towards a secure, usable, scientific computing environment.

### 3.1 Introduction

At the core of the Active Tables application is the Qengine kernel, a fortran application that provides the fundamental mechanisms for statistical analysis and the simultaneous solution of the thermochemical network. This analysis is performed on the basis of the underlying collection of thermochemical data. There are two kinds of data, one, historical (static) thermochemical network data in *Libraries* and, two, user created (dynamic) thermochemical network data inside user created *Experiments*. Some of the currently included Libraries are:

1. Main library (network data developed at Argonne)
2. CODATA library
3. JANAF library
4. Gurvich library

The Active Tables Web Service wraps up the Qengine kernel inside a Web Service interface so as to provide global access to the kernel. This wrapper service implements generic and extensible primitives providing for:

1. Security: for authentication, authorization and access control.
2. Experimental setup through user data manipulation.
3. Execution, query and analysis of experiments designed.

These primitives are described in more detail in Section 3.4.

The Active Tables Clients act as intuitive graphical user interfaces for the following:

1. Administration and setup of users and their permissions.
2. Setup of user experiments and associated data.
3. Viewing the functional relationships between the thermochemical data sets being worked with in a particular experiment.
4. Viewing results of an experiment, and export of the analysis thereof.

The various Active Tables client(s) as of the moment are:

1. Active Tables Web (Portlet) Client for the CMCS Portal.  
(production; in CVS)
2. Active Tables Web (Portlet) Client for Jetspeed.  
(beta; to be checked into the CVS)

3. Active Tables Java Swing Client.  
(alpha, prototype; to be check into the CVS)

Figure 3.1 describes the overall architecture of the Active Tables application, and it consists of the Active Tables Web Service, and the Active Tables Client, as shown.

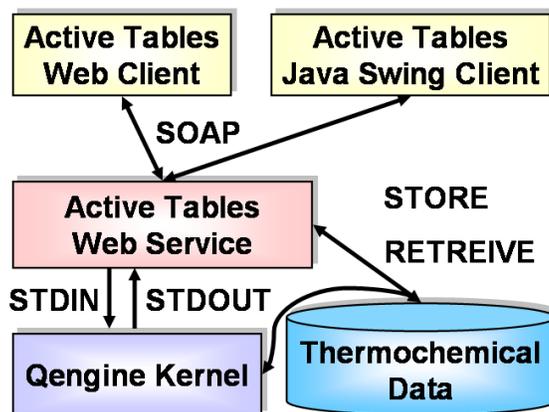


Figure 3.1: The overall Active Tables application Architecture

Section 3.2 describes briefly the background for the Qengine kernel from the security and integration point of view, followed by Section 3.3 describing scenarios of usage of the Active Tables application. Sections 3.4 and 3.5 describe the the design of the Active Tables Web Service and the Active Tables Client, respectively. This is followed by a discussion on the issues involved in the design and implementation of the Active Tables application, in Section 3.6.

## 3.2 Background

The Qengine kernel has a multitude of inbuilt features to support users, user to roles mapping, user data, standard library data, and access rights (be it user access or guest access) to these data.

The default startup of the Qengine kernel uses two files, namely Users.txt and Notes.txt to control the users, their roles, and the access permissions. The Users.txt files contents are detailed in Table 3.1, and it primarily consists of the mapping between users and their role, the username/password pair. The Notes.txt files contents are as detailed in Table 3.2, and it consists of data mapping between the username, and the user directory, and the third-party access password.

The Qengine kernel always looks for Users.txt and Notes.txt in the Qengine kernel install root directory, though it can also be started up with input parameters, such as 'Qengine.exe xyz'. With this input, the Qengine kernel looks for 'xyzUsers.txt' and 'xyzNotes.txt'. At the moment 'xyz' cannot be used to specify part of a file path, as the Qengine kernel validates the 'xyz' by checking whether the string 'xyz' conforms to the proper syntax for file names.

Briefly, the command line is:

```
C:\<install-directory>\Qengine.exe [String: like 'xyz']
```

Column	First	Second	Third
Column name	User type	User name	Password
Column type	Integer	String	String
Column values possible	Please see table 3.3.	Not case sensitive, but this is what is shown on screen, so should be readable.	Case sensitive.

Table 3.1: Detail of Users.txt

Column	First	Second	Third
Column name	User name	User directory	Password
Column type	String	String	String
Column values possible	Please see table 3.3.	Relative path to user experiment directory.	Case sensitive.

Table 3.2: Detail of Notes.txt

Column Value	Description
0	not authorized
1	guest
2	registered user
3	power user
4	admin
5	creator

Table 3.3: Column values for user type, in Users.txt

If 'xyz' contains only characters allowable for a file name (obeying both Win and Linux restrictions), the Qengine kernel expects files 'xyzUsers.txt' and 'xyzNotes.txt'. If there is something wrong with 'xyz' then it defaults to Users.txt and Notes.txt.

These files should have their full content, at least with respect to Libraries and Notes (and users who own them) that should be accessible during the launched session (best is to copy original Users.txt and Notes.txt and add the new user data).

Note:

1. The max length of 'xyzUsers.txt' and 'xyzNotes.txt' strings is currently set at 64 chars. That can be changed if needed; please contact Branko Ruscic (ruscic@anl.gov).
2. If User.txt/Notes.txt (or equivalent) are not found, the only possible user is "Guest".

### 3.2.1 Support for Dynamic Users

In its current form, the Users.txt and Notes.txt can be very well used to administer registered users, guests, administrators, etc and their data. There are a few downsides to this scheme, namely:

1. Working with user data in text files can become cumbersome.
2. Support for groups, and sessions, is currently not supported.

## 3.3 Scenarios of usage

All of the Active Tables client(s) access the Active Tables Web Service and work in the environments within which the client application is hosted, like a portal for the Active Tables Web (portlet) Client(s), and allow the user access to the basic query, solve and visualization features that the Active Tables Web Service provides.

At the moment the Active Tables Web Service works with a fixed set of users, and administration is hidden behind a manual edit of text files, and creation of user directories. This document describes the three scenarios of usage, envisioned, and further describes the use-cases for each scenario and the design involved.

The three scenarios of usage, can be represented through the workings of three classes of users, namely:

1. Administrator User
2. Registered User
3. Guest User

These users are described in detail, along with the use-cases envisioned for them, here.

### 3.3.1 Administrator User

This is a non-Qengine user, and is maintained by the Active Tables Web Service, through a separate mechanism (most probably through a database).

This user needs to have a strong username/password pair, which needs to be maintained in an encrypted manner. Also, connectivity between clients and the Active

Tables Web Service needs to be encrypted (https is a favored option). The issue here is of certificate management.

The Active Tables Web Service will be bootstrapped with one such Administrative user, who can then create others.

The envisioned use-cases for this user are:

1. Add Registered/Administrative User
2. Edit/Update Registered/Administrative User Data
  - (a) Change password
3. Delete Registered/Administrative User

### 3.3.2 Registered User

This is a, part, Qengine user, and a, part, Active Tables Web Service managed user. The user creation is done through the Administrator User. Editing of user-data and Deletion by both the Administrative user, and self.

The username/password/user-session-directories information is maintained in the Users.txt and Notes.txt. Sessions are maintained through entries in the Notes.txt.

Sharing of data is done through both the Notes.txt and an externally maintained Access Control List(s) (ACLs).

The envisioned use-cases for this user are:

1. Edit/Update Data
  - (a) Change password
2. Add/Use/Edit/Remove user objects.
  - (a) Objects are the user-session-directories.
3. Add/Edit/Remove accesses to user object(s), or maintain ACL for user(s).
4. Set currently used user-session-directory, whether be it a self-session, or a shared-session.
5. Query/Solve/Visualize: species, reactions, and networks.
6. Import/Export files xml/txt from and to the user-session-workspace.

### 3.3.3 Guest User

This is a guest user, and is a Qengine user. A special user, namely Active\_Tables\_Guest, is created and used only for the guest sessions.

The guest sessions are created and destroyed dynamically, and no data from the guest sessions is saved within the Active Tables application, or:

1. Guest sessions are created on the fly, when a user logs on as Active\_Tables\_Guest.
2. Guest sessions are destroyed on logout, timeout, and restart of the Active Tables Web Service.

Note: This user can access only certain areas, that are either:

1. Read-only access to the libraries

2. And to possibly shared folders, depending on the ACLs.

The envisioned use-cases for this user are:

1. Set currently used user-session-directory, whether be it a self-session, or a shared-session.
2. Query/Solve/Visualize: species, reactions, and networks.
3. Import/Export files xml/txt from and to the user-session-workspace.

### 3.4 Design of the Active Tables Web Service

The design of the Active Tables Web Service consists of two parts, the interface that the Web Service exposes and the implementation behind the interface. We first list out and describe the interface and then the design of the components behind the interface in a prototype implementation.

Figure 3.2 describes the architecture of the Active Tables Web Service, and it consists of the implemented interfaces, the request fulfilment engine, the authentication module, the filter module and the stores, as shown.

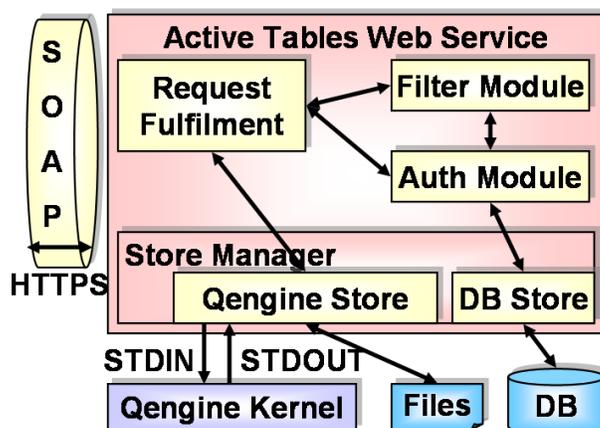


Figure 3.2: The Architecture of the Active Tables Web Service

#### 3.4.1 Interfaces

The design of the Active Tables Web Service consists of three interfaces, which are listed and described here:

1. Active Tables Authentication interface

This interface is used to enable interaction with the authentication mechanism supported.

```

(a) login(
    String username,
    String credential
)
    returns String token,
    throws Exception;
  
```

This method is used to authenticate to the Active Tables Web Service, and in return get an authentication token, which is to be used in further calls to the Active Tables Web Service.

At the moment the username is mapped with a password. This can be changed in the future to support arbitrary credentials.

```
(b) logout(  
    String token  
    )  
    return void,  
    throws Exception;
```

This method is used to reset the authentication status with the Active Tables Web Service, or to end a currently running authenticated session. This method has no return value.

## 2. Active Tables Data Manipulation interface

This interface enables manipulation of user data, that is managed by the Active Tables Web Service on behalf of the users, Administrative, Registered, and Guest.

This interface allows the Administrative user to add, update, delete and query Users, and their data, thus satisfying the necessary conditions of the respective use-cases.

This interface also allows the Registered and Guest users to manage the objects, and data that they have control and access over, and to interact with the kernel through the means of actions and queries on the results thereof.

```
(a) manipulateData(  
    String token,  
    String action-type,  
    String data-type,  
    String xml-rep-of-object  
    )  
    returns boolean status,  
    throws Exception;
```

This method is used to manipulate the data managed by the Active Tables Web Service. Some of the data include, User, User Objects (user sessions, session data), and User Object Access Control Lists.

The method requires the authentication token as part of the request, in order to authenticate the request and to recognize the user.

The action-type input is currently one of:

- i. Add.
- ii. Update.
- iii. Delete.

The object-type input is currently one of:

- i. User.
- ii. User Object.

iii. User Object Access.

iv. Qengine Data.

```
(b) queryData(  
    String token,  
    String query-type,  
    String object-type,  
    String xml-rep-of-query  
    )  
returns String xml-rep-of-object(s),  
throws Exception;
```

This method is used to query and retrieve the data managed by the Active Tables Web Service. The data being managed by the Active Tables Web Service as are as described above.

This method also requires the authentication token as part of the request.

The query-type input is currently one of:

i. Active Tables Web Service Data.

ii. Qengine Command.

iii. Import of User Data.

iv. Export of User Data.

v. Parsing of Network Solution.

### 3. Legacy Active Tables Data Manipulation interface

The functionality of this interface has been superseded by the above two interfaces, but it exists in the current production version of the Active Tables Web Service for backward compatibility with older versions of the Active Tables client(s).

```
(a) startExecuting(  
    void  
    )  
returns void,  
throws Exception;
```

This method is used to start executing an instance of the Qengine kernel, redirect its input/output streams, and tie it to the current session.

This functionality, of being able to “command” the execution of the Qengine kernel should not be public and should be controlled by a manager process. Keeping in this mind, these interfaces were re-designed.

```
(b) importNotes(  
    String fileUrl,  
    String fileName,  
    byte[] fileData,  
    String token  
    )  
returns void,
```

throws Exception;

This method is used to import user data, xml files, to the user-Notes directory on the Web Service.

The `fileUrl` input is the url of the originating position for the file, which is the location of the file on the DAV server for the Active Tables Web (Portlet) Client for the CMCS Portal, and the current `file://` path for the Active Tables Web (Portlet) Client for Jetspeed.

The `fileData` input is the file data as a base-64 encoded byte array.

The `token` input is the username for conducting the operation.

```
(c) exportNotes(  
    String token  
)  
    returns byte[] exportData,  
    throws Exception;
```

This method is used to export the user query data, as an xml file, to either the DAV server, in the Active Tables Web (Portlet) Client for the CMCS Portal, or to a temporary directory on the portal server (from where it can be downloaded) in the Active Tables Web (Portlet) Client for Jetspeed.

The `token` input is the username for conducting the operation.

The return is in the form of a base-64 encoded byte array, containing the encoded representation of the query data in an xml file.

```
(d) parse(  
    String token  
)  
    returns byte[] parseData,  
    throws Exception;
```

This method is used to run the AT&T DOT program, on the serialized contents of the Network Encyclopedia that the user had imported earlier.

The `token` input is the username for conducting the operation.

The return is in the form of a base-64 encoded byte array, containing the encoded representation of a zip file containing the graphical representation of the network.

```
(e) sendCommand(  
    String command-input,  
    String token  
)  
    returns String command-reply,  
    throws Exception;
```

This method is used to send a Qengine kernel command from the client(s) to the Active Tables Web Service.

The `command-input` input is the Qengine kernel command in the form of a text string.

The token input is the username for conducting the operation.

The return is in the form of a text string, containing the, possibly formatted and processed, output from the command execution on the Qengine kernel.

### 3.4.2 Implementation

The implementation of the Active Tables Web Service consists of three modules, namely, the:

1. **Request Fulfilment Engine:** This module is responsible for the implementing the interfaces and providing for the services requested. The design of the interfaces was envisioned such that the input would be in the form of a xml representation of objects, which would then be converted from the xml form to the java objects and then be easily manipulated.

Some of the issues and thoughts with regards to this module are listed here:

- Currently Castor and Jaxb are two technologies that are working towards java-to-xml binding.

More information can be found at the Castor, or Jaxb sites:

<http://castor.exolab.org/>

<http://java.sun.com/xml/jaxb/>

- For working with XML, XML-Schema's need to be formulated, along with "description" xml files that help in the java  $\rightarrow$  xml, and vice-versa, translations.
2. **Authentication and Filter:** These modules are responsible for implementing the authentication mapping between the username/credential duo to the active-session token, on one hand and to filter (ie apply the authorizations and access mechanisms to) the requests from the client for either manipulation or query of the data.

A typical scenario for the authentication module would be during the login of the user, thus creating the user session token, the authentication of token per request, and the further destruction of the token on logout or time-out.

A typical scenario for the Filter module would be to work with access requests of a particular kind (ADD, UPDATE, DELETE, QUERY, etc) for a particular object (USER, USER-OBJECT, etc) and given the current users role, and the access restrictions defined on the object, determine whether the request is allowed or not.

3. **Stores:** The stores implement the Store interface, which consists of a few simple primitives, as listed here:

- `createObject(`  
    Integer object-type,  
    Attribute[] attributes  
    )  
    returns object-id,  
    throws Exception;
- `addObjectAttributes(`  
    String object-id,

```

Attribute[] attributes
)
returns Boolean,
throws Exception;

• queryObjectById(
String id
)
returns xml-rep-of-object(s),
throws Exception;

• queryObjectByAttribute(
Attribute attribute
)
returns xml-rep-of-object(s),
throws Exception;

• updateObject(
String object-id,
Attribute[] attributes
)
returns Boolean,
throws Exception;

• deleteObject(
String object-id,
)
returns Boolean,
throws Exception;

```

The Attribute object consists of a, String based, NAME and VALUE pair that represent the various data that the object owns.

The implementation of the Qengine Store and the DB Store, as shown in Figure 3.2 can implement helper functions, that internally use the store interfaces to provide higher level services.

### 3.5 Design of the Active Tables Client

The major points to be taken into consideration when designing and writing out the Active Tables Client(s) are:

- Maintain session state, ie username/credential to security-token mapping.
- Usability, through the use of menus, icons, shortcuts, keyboard MNEMONICS.
- Input validation (to a certain degree).
- Functionally sound and reuse of code (major point) between various clients.

### 3.6 Design Issues

In this section are described the various design issues, especially to do with extensibility and security.

### 3.6.1 Extensibility

- Generic but usable interfaces for extensibility.
- Using store managers and stores so that implementation can be changed.
- Using a generic object and attributes based store to increase extensibility at a certain programming cost.

### 3.6.2 Security

Some of the common security issues are listed here with a discussion on methods of securing them.

#### Client

1. Checks on maximum "allowable" input size should be enforced, as this is the most common source of web-application attacks, form-based buffer overflow.
2. Connections between clients and the Active Tables Web Service, especially with regards to the Administrative/Login operations, should be done through HTTPS. This issue then falls back to certificate issuance/management.

#### Web Service

1. Axis ?wsdl service is something that can be a potential threat! This is about security in anonymity; as when the outsiders dont know about the interface that is exposed through Axis, it provides us with some security, as opposed to when the wsdl is exposed, and the outsider knows exactly what the api is, and can try to break it.

Keeping this in mind, the interfaces for the new Web Service have been designed keeping in mind simplicity and usability, but at the same time making sure that as little information is given out with regards to the internal application structure, sequence flow, and such issues.

2. The issue of management of users within the Tomcat web-application container and its "admin" and "manager" web-applications, needs to be looked into, but is beyond the scope of this document.
3. Connections between clients and the Active Tables Web Service, especially with regards to the Administrative/Login operations, should be done through HTTPS. This issue then falls back to certificate issuance/management.
4. Strong administrator passwords are a must for the protection of the application. In the future quite possibly a configurable password "validation" module will be required.

Currently the implementation can use simple username/password paris to handler authentication and a auto-generated token for authorization, but this should be upgraded to username/credential mapping and secure connections for higher security.

5. Mechanisms should be put into place for handling DOS attacks, either through input buffer overflow, and other such attacks. Although not much can be done from the point of view of the web-service, apart from input validation,

but most of this should be features that the web-application environment that axis is running within, and of course within axis.

Not performing good and through testing is one of the biggest security loop-holes for web-applications and should be avoided. This requires good application design, understanding of the web-application hosting environment, and comprehensive testing plans, and execution thereof.

#### Qengine kernel

1. Each command tried out by the user will have to be parsed through for sensitive words like "set", "lib", "execute", and others for potential commands that the user is trying to run on the Qengine.exe. As, if security is to be maintained then these commands should go through the Active Tables Web Service.

Keeping in mind this, the Filter Module has been envisioned, which would take care of filtering the command sequences, depending on the role assigned to the current user.

---

## 4 Development

---

### 4.1 Notes

This section describes the various notes that the developer should take care of when working with and developing additional functionality around the Qengine kernel, and the Active Tables application.

General notes are in Sub-Section 4.1.1 followed by CMCS Portal Development notes in 4.1.3, and lastly specific notes in Sub-Section 4.1.2.

#### 4.1.1 General

- For the CMCS portal, sessions are tracked using browser memory cookies, which are not stored on the disk anywhere. Thus, if you close the browser, your session is gone. Two *instances* of IE have different memory (obviously), so you can have two different sessions using these instances. These may be for the same user or different user.
- The Qengine kernel does not write to any files outside of the User Notes folder.
- We dont have any skeleton for the web service. We directly work with Impl. This is because we want Impl to implement HttpSessionBindindListener, so that at the start of the session, it can create new users and at the end destroy those users. Now if we want skeleton, the skeleton will have to implement HttpSessionBindindListener. Since the skeleton is auto-generated every time the WSDL changes, this is not convenient.
- The logic behind the 'set java' (A Qengine kernel command) is a hack, with Branko Ruscic (ruscic@anl.gov) help, to get around new-line. (Its also a way for us to customize the behavior of the Qengine kernel, so that we have an easier time in parsing the output).
- HTTP Sessions concepts:
  1. After logout, the session is not invalidated immediately. It is invalidated after the session timeout reaches (SessionListerner works. Verified) OR if the user logs back in with the same cookie, whichever is earlier. The Jetspeed user logged in may be different (SessionListener works. Verified for IE and Netscape).
  2. The auto-update of Chef causes sessions not to timeout while the user is logged in. But if the window is closed, session times out. In case of session timeout, SessionListener works perfectly; Verified.
  3. SessionListener is not called on Tomcat shutdown. Dont know why! But this is acceptable. There will be a few directories lying around, which is OK. Even if later there is a session with the same ID, its files will overwrite the previous files created. So does not matter.

## 4.1.2 AJmenu

The rationale behind choosing AJmenu, the advantages:

1. Free. For the license please see [1.3.2](#).
2. Provides multi-level hierarchies for free (many only provide a single-level for free, and its not advisable to hack the javascript code in a DOE project)
3. Works for many browsers. For more information please check the AJmenu site:  
<http://navsurf.com/dhtml/ajmenu/>.
4. Has a GUI-based menu builder
5. Looks good

The dis-advantages:

1. It is dynamic JavaScript, so depends on the other scrips that have been loaded by the portal page, and the rendering engine on the browser. Though it generally works fine with Netscape 7.02, and IE 6.0+, and dynamic scripting problems would occur whether or not AJmenu was used.

To make changes to the menu is fairly easy. The 'ActiveTablesQueryMenu.js' file can be edited in any text editor to create and enhance the menu. The Menu consists of the following components:

1. mainMenu: This object defines within the AJmenu environment a new menu bar, all that needs to be done is the initialization of this object through the mainMenu() method, that takes eight inputs, namely:

First Layout of the menu; either horizontal or vertical.

Second Spacing between menu's on the menu-bar, in px.

Third Background color while normal (ie, unselected).

Fourth Background color while highlited (ie, selected).

Fifth Font color while normal (ie, unselected).

Sixth Font color while highlited (ie, selected).

Seventh Background image if needed; else leave empty ("").

Eighth Menu-bar direction; either right or left.

2. subMenu: This object defines within the AJmenu environment a new menu, that can reside within the menuBar, and all that needs to be done is the initializationof this object through the subMenu() method, that takes five inputs, namely:

First Menu number, actually a string, of the form: starts with 1 and increment by 1, but for sub-menu(s) this number is preceeded by its parent lineage interspersed by 'i's.

Thus for a menu in the third-item place of the second menu on the menu bar, this number would be 2i3.

Second Background color while normal (ie, unselected).

Third Background color while highlited (ie, selected).

Fourth Font color while normal (ie, unselected).

Fifth Font color while highlighted (ie, selected).

3. menuItem: This represents an item on either the menuBar or the subMenu, which can be selected to produce the appropriate GUI effect. These objects are automatically created using the addItem() method, on either the menuBar or the subMenu objects. The addItem() method takes two inputs, namely:

First The label that is shown for that menu item.

Second The action that will be executed for that menu item. In our case we would, most probably, need to call an external javascript function; which would then set something in a form and then submit that form.

For creating a functional menu would require, at least, one menu-bar (ie, one mainMenu() call), one or more menu's (ie, one subMenu() call for each menu) and chosen number of menu-items added to the appropriate menu.

To change the AJmenu, there are two options:

1. Implement a custom JavaScript menu.
2. Implement another form of input that does not rely on the JavaScript menu. Available options are checkboxes, links, radio-buttons, Java applets, etc.

#### 4.1.3 CMCS Portal development

1. Information on the CMCS portal can be found at the CMCS project website: [urlhttp://cmcs.ca.sandia.gov/](http://cmcs.ca.sandia.gov/)
2. A short list of the contacts is available in Table 4.3. For more information, please check the CMCS website, or contact any of the Argonne National Laboratory contacts, listed in Table 4.1.
3. Accessing information/code:
  - **CVS:** More information on accessing the CMCS code from their CVS can be found in a document in the CVS, in the `cmcs/documentation/developer/final_cvs_steps.doc` file.

Information is also available here:

<http://cmcs.ca.sandia.gov/cvs.php>

- **portals, SAM,** : For accessing the portal(s) user accounts need to be created on the respective portals:
  - (a) Production: <http://cmcs.ca.sandia.gov:10081/cmcs/portal>
  - (b) Dev: <http://cmcs-dev.ca.sandia.gov:10081/cmcs/portal>
  - (c) Laptop: <http://cmcs-laptop.ca.sandia.gov:10081/cmcs/portal>

This can be done using the “Create New Account” functionality available on the, respective, portal.

The current convention for the passwords is to pre-pend the word “cmcs” by the first four alphabets of the username chosen. **Note:** This was a strategy chosen for ease of development and testing for before SC, and should be discontinued.

- **Team Calendar, DocuShare:** The CMCS team uses DocuShare, and a shared Team Calendar which requires additional access. Please have a look at the information available here:  
<http://scidac.ca.sandia.gov/View/Collection-10/>
  - **Team Mailing List:** Please check the following link for information:  
<http://cmcs.ca.sandia.gov/mailing.php>
  - **Telephonic Conferences, and VNC:** Information regarding access to Telephonic Conferences and VNC is usually found in the team meeting emails.
4. Building and testing the CMCS code and checking binaries into the CMCS CVS is described here:
- Checkout the CMCS code from the CVS.
  - Set the various ant “build.properties” as:
    - (a) Copy and modify the `deploy.cmcs-dev.ca.sandia.gov` to `deploy.[your machine]`.
    - (b) Set the properties: `portal.server.name`, and `portal.server.path.chef`, to the portal server being used.
    - (c) If the “deployPortal” Ant target is being used then set the properties: `portal.home`, `portal.data.home`, and `portal.build.dir`.
    - (d) Run
 

```
ant -propertyfile deploy.[your machine]
serverproperties
```
  - Build the CMCS portal by running the appropriate Ant target, such as:  
`ant [builddeployPortal]`—
  - Build the Active Tables code, by pointing to the CMCS portal directory in the “`deploy.dir`” property in the `project.properties`, of the `cmcs/src/portlet-cmcs` dir.
  - Test the portlet within the CMCS portal, on a local installation, with the Active Tables Web Service also running locally.
  - Once all the changes are satisfactorily tested, check in the changed code, ie jar files, etc in to the CMCS portal CVS checkout directories, and commit them, with a useful label.
5. Emails concerning the CMCS portal development effort, alongwith the Active Tables effort from the Argonne National Laboratory are available in the CVS, in the `cmcs/documentation/developer/emails/cmcs-emails.pst` file.

## 4.2 How To's

To change the interface of the web service, all in the `src/web-service` directory of the source distribution:

1. Rename the `Impl` file as a backup.
2. Run `'ant precompile'`, to generate the stubs and service classes.
3. Remove the generated `Impl` file.

4. Rename the original Impl back, and implement the changes, if any, to the api.
5. Remove the class files, just to be sure.
6. Run 'ant build', to compile.
7. Run 'ant deploy', to deploy the jar file. For this set *deploy.dir*, and *catalina.dir*.
8. Run 'ant deploy\_AT2', to deploy the Web Service. For this set *axis.http.url*.
9. Start the web-application container.
10. Check the auto generated WSDL to see if it gets generated, and also if the changes have taken place.  
`http://localhost:<port>/axis/services/<servicename>?wsdl`

#### 4.2.1 Testing

This section discusses the ways to test the Active Tables application during development, and testing phases.

This section consists of the following types of testing:

- Web Service
- Client
- Scenarios of testing
- Common mistakes and their solutions

TODO

#### 4.3 Contacts

The Argonne National Laboratory and Java CoG Kit contributors to the Active Tables project are listed in Table 4.1. Past team members are listed in Table 4.2. The contributors/contacts from the CMCS project are listed in Table 4.3.

Table 4.1: Argonne National Laboratory, Contacts

Name	e-mail	Divison	Role	Phone
Albert Wagner	wagner@tcg.anl.gov	CHEM	PI	-
Branko Ruscic	ruscic@anl.gov	CHEM	PI	-
Reinhardt E Pinzon	pinzon@chm.anl.gov	CHEM		-
Mike Minkoff	minkoff@mcs.anl.gov	MCS	PI	-
Gregor von Laszewski	gregor@mcs.anl.gov	MCS	PI	-
Sandra Bittner	bittner@mcs.anl.gov	MCS	Sysadmin	-

Table 4.2: Past Members Argonne National Laboratory, Contacts

Name	e-mail	Divison	Role	Phone
Patrick Wagstrom		CMU MCS	summer student	-
Shriram Krishnan		Indiana U.	summer student	-
Mihael Hategan	hategan@mcs.anl.gov	MCS	Developer	-
Shashank Shankar		MCS	Student	-
Sandeep Nijasure		MCS	Student	-
Kaizar Amin	amin@mcs.anl.gov	MCS	Student	-

Table 4.3: CMCS Project, Contacts

Name	e-mail	From	Phone
Larry Rahn	rahn@sandia.gov	SANDIA	-
David Leahy	djleahy@sandia.gov	SANDIA	-
Christine Yang	clyang@ca.sandia.gov	SANDIA	-
Carmen Pancarella	carmen@ca.sandia.gov	SANDIA	-
James D Myers	jim.myers@pnl.gov	PNL	-
Brett T Didier	brett.didier@pnl.gov	PNL	-
Karen L Schuchardt	Karen.Schuchardt@pnl.gov	PNL	-
Carina S Lansing	Carina.Lansing@pnl.gov	PNL	-
Theresa L Windus	Theresa.Windus@pnl.gov	PNL	-
Bill Pitz	pitz1@llnl.gov	LLNL	-
David Montoya	dmont@lanl.gov	LANL	-
Tom Allison	thomas.allison@nist.gov	NIST	-
Bill Green	whgreen@mit.edu	MIT	-
Michael Frenklach	myf@me.berkeley.edu	UCB	-

---

## 5 Installation

---

The scenarios to build, test and deploy the Active Tables components are described here.

The ATcT Web Service (WS) runs on the Windows box, and the development environment is setup through cygwin. The CMCS-Portal is build on Linux/or through Cygwin from their source tree.

The other option that can be used to host the Active Tables Web (Portlet) Client , is to use the Jetspeed portlet container.

A scenario that we follow to do a build is to compile and jar the activeTablesPortlet.jar and activeTablesService.jar and then copy them to the, respective WS and the Portal (be it either CMCS or Jetspeed), WEB-INF/lib directories.

So for the internal build, we have to modify the ATables2.wsdl to point the WS-client to the WS-server. Compile, deploy the service, test the functionalities required.

Once the WS-client and servers have been properly debugged then they can be deployed on the production machines, and then tested through the testing CMCS/Jetspeed-Portal machines.

Once the CMCS-Portal build checks out okay, the activeTables jar files can then be committed into the CMCS cvs srouce.

What follows is more detailed methodology for each step.

### 5.1 Quick Install guide

This quick install guide will describe how a fast configure and install of the Active Tables Web Service, and the Active Tables Web (Portlet) Client for Jetspeed can be done.

1. Get Java installed.
  - Need to set JAVA\_HOME, to the Java install directory.
  - Add Java\_HOME/bin to the sysetm PATH.
2. Get Ant installed.
  - Set ANT\_HOME, to the Ant install directory.
  - Add ANT\_HOME/bin to the system PATH.
3. Get Jakarta-Tomcat installed.
  - Set CATALINA\_HOME, to the Tomcat install directory.
  - Add CATALINA\_HOME/bin to the system PATH.
4. Get Apache-Axis installed into Tomcat.

- Copy the axis.war into the CATALINA\_HOME/webapps/ directory.
5. Get Apache-Jetspeed installed into Tomcat.
    - Copying the jetspeed.war into the CATALINA\_HOME/webapps/ directory.
  6. Download Active Tables application sources/binary distribution.
    - `cvs -d:pserver:anonymous@cvs.globus.org:/home/dsl/gregor/cmcs checkout -r atct-1-2-alpha cmcs`
  7. Configure and deploy the Active Tables Web Service:

- Edit the `src/web-service/etc/active-tables-web-service.properties`, and set the locations to the Qengine kernel, the AT&T Dot executable, and the xslt translators that you are using.

For a default installation within cygwin, these urls would look like:

```
qengine.install.dir=c:\\cygwin\\usr\\local\\bin\\qengine
graphviz.install.dir=c:\\cygwin\\usr\\local\\bin\\graphviz
xslt.install.dir=c:\\cygwin\\usr\\local\\bin\\xslt
```

- Edit the `src/web-service/project.properties`, and set the locations to the `servlet.jar` in within Tomcat, the Axis web-application directory (as the deploy directory), and the axis url for the url to the axis servlet.

For a default installation within cygwin, these paths/urls would look like:

```
servlet.lib.dir=c:\\cygwin\\usr\\local\\bin\\jakarta-tomcat-4.1.24\\common\\lib
deploy.dir=c:\\cygwin\\usr\\local\\bin\\jakarta-tomcat-4.1.24\\webapps\\axis
axis.url=http://<ip-address>:8080/axis/servlet/AxisServlet
```

- Run 'ant' to build and deploy the jar files. The 'deploy.jar' target will build and deploy the jar files, and the 'deploy\_AT2' target will deploy the service into Axis.

*Note:* You might need to restart Tomcat/Axis, after the 'deploy.jar' step.

- Restart Tomcat and verify that the wsdl of the deployed service can be viewed.

8. Edit the `src/portlet-jetspeed/etc/active-tables-client.properties`, and set the `active-tables.web-service.url` to the Web Service url that you are using.

- For a default installation of Tomcat and Axis, the url would look like: `http://<ip-address>:8080/axis/services/ATables2`

- Edit the `src/portlet-jetspeed/project.properties`, and set the locations to the `servlet.jar` in within Tomcat, and the Jetspeed web-application directory (as the deploy directory).

For a default installation within cygwin, these paths/urls would look like:

```
servlet.lib.dir=c:\\cygwin\\usr\\local\\bin\\jakarta-tomcat-4.1.24\\common\\lib
```

```
deploy.dir=c:\\cygwin\\usr\\local\\
bin\\jakarta-tomcat-4.1.24\\webapps\\jetspeed
```

- Run 'ant' to build and deploy the jar files.
- Restart Tomcat, and use the portal administration features of Jetspeed to add the Active Tables Web (Portal) Client for Jetspeed to the selected portlets and save your selections. Verify that you can use the portlet to access the Active Tables Web Service.

9.

## 5.2 Prerequisites

In order to use the Active Tables application, the Java Runtime Environment version 1.4, available from the Java Web-Site is required. Additionally, if you plan to compile the ATcT application from sources, you will need the full Java Development Kit, version 1.4, available from the same web-site, and a recent version of the Apache Ant build system. To deploy and use the ATcT Web Service, you will need the SOAP implementation from Apache, AXIS.

To install and use the CMCS based client, you will need the CMCS sources. More information can be found on this site:

<http://www.cmcs.org/>

To install and use the Jetspeed client, you will need the Jetspeed portlet container implementation.

At this time we recommend to use the following packages, as we have not yet tested ATcT application with any other Java version.

1. **JDK 1.4.1.02:** The Active Tables Web Service and the client(s) all need the Java runtime environment, that can be downloaded from:  
<http://java.sun.com>
2. **Apache Ant-1.5.3:** Build from sources of the ATcT Application requires Apache Ant, that can be downloaded from:  
<http://ant.apache.org>
3. **Jakarta Tomcat 4.1.x:** Deployment of the web-applications requires a web-application container, and we have tested on Jakarta Tomcat 4.1.x, that can be downloaded from:  
<http://jakarta.apache.org/tomcat/>
4. **Apache Axis 1.1rc2:** Deployment of the Active Tables Web Service requires the Apache Axis web-application, that can be downloaded from:  
<http://ws.apache.org/axis/>
5. **Jakarta Jetspeed 1.4b3:** Deployment of the Active Tables Web (Portlet) client for Jetspeed requires the Apache Jetspeed web-application, that can be downloaded from:  
<http://jakarta.apache.org/jetspeed/>

### 5.2.1 What does the Web Service need to run?

1. Java (platform).

2. Jakarta-tomcat with Apache-axis 1.1 RC2 (web-server, servlet-container, and SOAP implementation app).
3. Ant (to compile and deploy).
4. ATcT Web Service jar file (or class files).
  - (a) The CVS root is: WS\_CVS
 

```
      :ext:<username>@<machine>:/home/dsl/gregor/cmcs
```
  - (b) The module name is cmcs.
5. ATcT schemas jar file (or class files) From WS\_CVS.
6. Qengine kernel version 2, from Branko Ruscic (ruscic@anl.gov).
7. GraphViz: from AT&T (for generation of network).
8. cygwin (for CVS and path to QEngine.exe).
  - (a) To be installed in the root directory:
 

```
      C:\cygwin\
```
  - (b) And user directory by the name of Administrator created:
 

```
      C:\cygwin\home\Administrator\.
```
  - (c) This is where the CVS source is to be checked out, and through where the Web Service will find the Qengine.exe.
9. Everything on a Windows 2000+ machine.

### 5.2.2 What does the Portlet need to run?

1. Java (platform).
2. Jakarta-tomcat (web server and servlet-container).
3. Apache-Jakarta-Jetspeed (Enterprise Information Portal webapp).
4. Ant (to compile and deploy).
5. CMCS Portlet source from PORTAL\_CVS, and ATcT Portlet from WS\_CVS.
  - (a) The CVS root is: PORTAL\_CVS
 

```
      scidac.ca.sandia.gov:/data/cvs/cmcs
```
  - (b) The module name is cmcs.
  - (c) Work with the cmcs branch.
  - (d) Example: `cvs co cmcs; cvs update -dP cmcs`
6. Everything on a Linux machine.
7. Locations of ATcT stuff in PORTAL\_CVS hierarchy: (Relative to cmcs/portal/webapps where "cmcs" is the directory for your checkout)
 

Note: All the files are prefixed with ( ActiveTables — activeTables — ATcT )

  - (a) Template files: WEB-INF/cmcs/templates/vm/portlets/html/
  - (b) Xreg file: WEB-INF/conf

- (c) Jar files: WEB-INF/lib
- (d) JavaScript files for CSS: js/
- (e) Images: images/
- (f) Html: html/ActiveTables/

## 5.3 Download

There are two mechanisms of obtaining the Active Tables application releases, as source code through the CVS, and as pre-built release packages from the CMCS project page, which can be found here:

<http://www.globus.org/cog/projects/cmcs>

### 5.3.1 CVS Release Tags

At the time of writing of this document there were two release(s) made. The first one is labeled atct-1-1, and is compliant with the version 1.1 release of the CMCS, and the next version is labeled atct-1-2alpha, and it has significant changes to the previous version. It is recommended to use the latest version, and effort has been made to keep the new version as backward compatible as possible.

### 5.3.2 Downloading the binary distribution

TODO

### 5.3.3 Downloading the sources

Access to the Active Tables code is granted through the “cmcs” module. This is in a CVS repository that is accessible only through the Argonne National Laboratory internal network. To get access to this repository, please contact Gregor von Laszewski (gregor@mcs.anl.gov).

Once access is granted, the module can be checked out via:

```
cvs -d:pserver:<username>@cvs.globus.org:/home/dsl/gregor/cmcs  
checkout -r atct-1-2-alpha cmcs
```

## 5.4 Building from the sources

This section describes the mechanisms to build the various parts of the Active Tables application from the sources.

### 5.4.1 Building the ActiveTablesQueryExport.jar

1. Checkout the cmcs module from WS\_CVS.
2. Follow the directions given in the CMCS\_Description\_of\_xsds.ppt file in the cvs.

## 5.4.2 Building the Active Tables Web Service

### 1. Presetting:

#### (a) Cygwin installed at

C:\cygwin

#### i. User directory created at

C:\cygwin\home\Administrator\.

#### (b) Web Services container, axis installed.

#### i. AXIS\_HOME environment variable, in cygwin, set to point to where axis is installed.

#### ii. AXIS\_HOME translated using the cygwin cygpath program, as:

```
AXIS\_HOME=/cygdrive/c/jakarta-tomcat-4.1.24/webapps/axis
export AXIS\_HOME
AXIS\_HOME=`cygpath --path --windows "$AXIS\_HOME"`
export AXIS\_HOME
```

#### (c) ActiveTablesQueryExport.jar installed in the WS\_CVS/cmcs/src/java/lib directory.

### 2. Checkout the cmcs module from WS\_CVS, in the directory:

C:\cygwin\home\Administrator

### 3. Put the desired IP address and port number for the service hosting in the WSDL file, in WS\_CVS/cmcs/src/ATables2.wsdl, and make the appropriate changes to the axis.url in the WS\_CVS/cmcs/src/java/build.xml file.

#### (a) Note: Mentioning "localhost" in WSDL does NOT work.

#### (b) Also, as mentioned before, the portal needs to know about the location of the web service. Thus, whenever the IP address of the web service machine changes, you need to perform this step, regenerate the stubs, and jar, finally copying the activeTablesService.jar into the portal distribution.

#### i. It would be better to have a configuration value in the ActiveTablesQuery.xreg portlet configuration file.

### 4. Extract the QEngine, the ATcT Fortran application, by un-tarring QEngine.tar.

#### (a) Example:

```
cd WS\_CVS/cmcs/src/QEngine/version2/ATcT/Qengine/Debug;
tar -xvf QEngine.tar
```

#### (b) Note: By default the Web Service looks for the QEngine.exe executable in the directory:

```
C:\cygwin\home\Administrator\cmcs\src\QEngine\
version2\ATcT\Qengine\Debug\
```

### 5. Configure the Active Tables Web Service:

- Edit the src/web-service/etc/active-tables-web-service.properties, and set the locations to the Qengine kernel, the AT&T Dot executable, and the xslt translators that you are using.

For a default installation within cygwin, these urls would look like:

```
qengine.install.dir=c:\\cygwin\\usr\\local\\bin\\qengine
graphviz.install.dir=c:\\cygwin\\usr\\local\\bin\\graphviz
xslt.install.dir=c:\\cygwin\\usr\\local\\bin\\xslt
```

- Edit the `src/web-service/project.properties`, and set the locations to the `servlet.jar` in within Tomcat, the Axis web-application directory (as the deploy directory), and the axis url for the url to the axis servlet.

For a default installation within cygwin, these paths/urls would look like:

```
servlet.lib.dir=c:\\cygwin\\usr\\local\\
bin\\jakarta-tomcat-4.1.24\\common\\lib
deploy.dir=c:\\cygwin\\usr\\local\\
bin\\jakarta-tomcat-4.1.24\\webapps\\axis
axis.url=http://<ip-address>:8080/axis/servlet/AxisServlet
```

6. Build the Web Service code by running ant, with the default target, from the `WS_CVS/cmcs/src/java/` directory.

- (a) Run 'ant' to build and deploy the jar files. The 'deploy\_jar' target will build and deploy the jar files, and the 'deploy\_AT2' target will deploy the service into Axis.

*Note:* You might need to restart Tomcat/Axis, after the 'deploy\_jar' step.

- (b) Note: You may have to undeploy the previous deploy and that can be easily done by running ant, with the `undeploy_AT2` target.
- (c) Note: The Web Application container, and axis need to be running for the [un-]deploy to work.
- (d) Restart Tomcat and verify that the wsdl of the deployed service can be viewed.
- (e) Help for `org.apache.axis.tools.ant.wsdl.Wsdl2javaAntTask` at <http://venus.math.klte.hu/docs/apache/axis/ant/axis-wsdl2java.html>

### 5.4.3 Compiling the Web (Portlet) Client for the CMCS portal

1. Presetting:

- (a) JSP/Servlet container Jakarta-tomcat installed
  - i. `CATALINA_HOME` environment variable, in cygwin, set to point to where tomcat is installed.
  - ii. `JETSPEED_HOME` environment variable, in cygwin, set to point to where Jakarta-Jetspeed is installed.

2. Check out the cmcs module from `PORTAL_CVS`.

3. Edit the `src/portlet-jetspeed/etc/active-tables-client.properties`, and set the `active-tables.web-service.url` to the Web Service url that you are using.

- For a default installation of Tomcat and Axis, the url would look like:  
`http://<ip-address>:8080/axis/services/ATables2`

- Edit the `src/portlet-jetspeed/project.properties`, and set the locations to the `servlet.jar` in within Tomcat, and the Jetspeed web-application directory (as the deploy directory).

For a default installation within cygwin, these paths/urls would look like:

```
servlet.lib.dir=c:\\cygwin\\usr\\local\\
bin\\jakarta-tomcat-4.1.24\\common\\lib
deploy.dir=c:\\cygwin\\usr\\local\\
bin\\jakarta-tomcat-4.1.24\\webapps\\jetspeed
```

- Run 'ant' to build and deploy the jar files.
- Restart Tomcat, and use the portal administration features of Jetspeed to add the Active Tables Web (Portal) Client for Jetspeed to the selected portlets and save your selections. Verify that you can use the portlet to access the Active Tables Web Service.

#### 5.4.4 Compiling the Web (Portlet) Client for Jetspeed

NOTE: This section is to be completed when the client gets written.

### 5.5 Deployment and Administration

Herein are listed some for the issues relate to the Deployment and Administration of the Active Tables application.

#### 5.5.1 Issues

This is a checklist of things to worry about when deploying the Active Tables Web Service.

1. HTTPS connectivity between Active Tables Client(s) and the Active Tables Web Service.
2. Administration of user accounts, and programs installed, and access to resources being used for the deployment.
3. System security through firewalls, software system patches and reboots, and system monitoring through a logging and reporting infrastructure.
4. Administration of Active Tables application specific users, and their data.

#### 5.5.2 Public Web Service on Bubbles

This is described in the document available at this location:

[http://www-i-unix.mcs.anl.gov/~laszewsk/shankar/doc/Managing\\_the\\_Active\\_Tables\\_Web\\_Service.doc](http://www-i-unix.mcs.anl.gov/~laszewsk/shankar/doc/Managing_the_Active_Tables_Web_Service.doc).

(Please note: this is an Argonne internal network accessible link only.)

---

## 6 Usage

---

### 6.1 Active Tables Web (Portlet) Client

The usage of the Active Tables application is characterized by the usage of the Active Tables Web (Portlet) Client for the CMCS Portal, as shown in Sub-section 6.1.1. The functionality and the major look/feel of the Active Tables Web (Portlet) Client for Jetspeed is very similar.

#### 6.1.1 Active Tables Web (Portlet) Client for the CMCS Portal

This tutorial will introduce the basic functioning of the Active Thermochemical Tables portlet. This portlet will be referred to as the ATcT portlet from this point on. The ATcT portlet is available under the 'Active Tables Developers' Workspace, for demo purposes through the demo login.

For more information on the CMCS portal, workspaces, account creation and other features please refer to the respective tutorials.

#### Initial Portal Page and ATcT Portlet

The ATcT portlet landing page is the initial web page that the user encounters when the 'Active Tables' workspace is chosen, and the 'ATcT' Tab on the left column is chosen. This is shown in Figure 6.1.

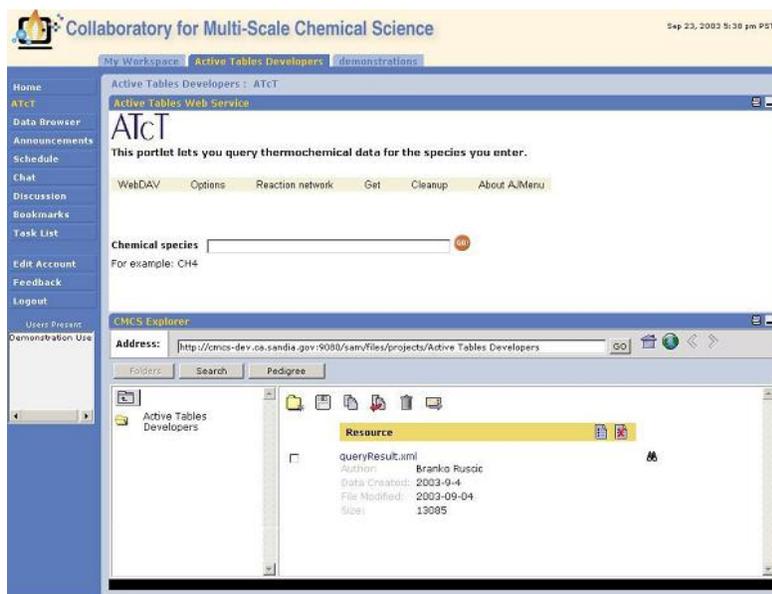


Figure 6.1: CMCS Portal Initial Web Page

The portlet on the top, in Figure 6.1, is the ATcT portlet, and the one on the bottom is the CMCS Explorer, the CMCS Data management portlet. More information about the CMCS Explorer, refer to the CMCS Data Management tutorial.

*Note:* The import/export functionalities of the ATcT portlet will not function correctly if the CMCS Explorer portlet is not used along-with it.

The ATcT portlet, shown in Figure 6.2, is composed of three areas:

1. The Menu
2. The Command Box
3. The Data Area

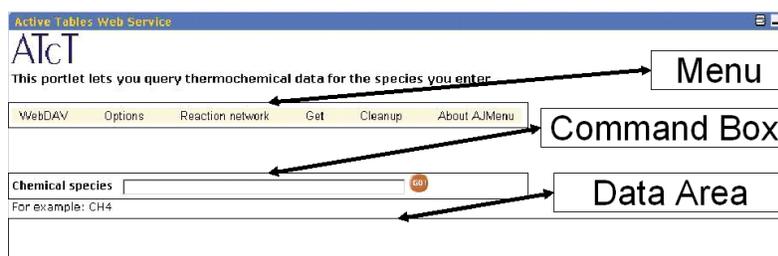


Figure 6.2: Active Tables Web (Portlet) Client for the CMCS Portal

The Menu and the Command Box are the primary interaction mechanisms to the ATcT kernel, with which the ATcT portlet interacts, to enable solving Thermochemical Tables. Interaction through pre-programmed commands is through the Menu, whereas the Command Box can be used for almost any kind of direct interaction with the ATcT kernel. The results of commands given, from either the Menu, or the Command Box are shown in the Data Area of the ATcT portlet.

#### Detail and working with the Menu

The Menu has the following hierarchy:

- **WebDAV:** This menu item links to the DAV functionality like export and import.

*Note:* As mentioned previously the DAV functionality will not work as expected if the CMCS Explorer portlet is not activated in the same session.

- **Import files from DAV Server:** This command is used to import files from the DAV server into the user space at on the ATcT kernel. These files can then be used as the inputs for the various reactions, and networks, which can then be 'solve'd for.

On selecting this option, shown in 6.3 the portlet displays the a message stating that the desired files to be selected in the CMCS Explorer, as shown in 6.4.

Clicking on the 'Import files' button, shown in Figure 6.4 will then instruct the ATcT portlet to import files from the DAV server, and it notifies of the success or failure of the import(s) operation, as shown in Figure 6.5.

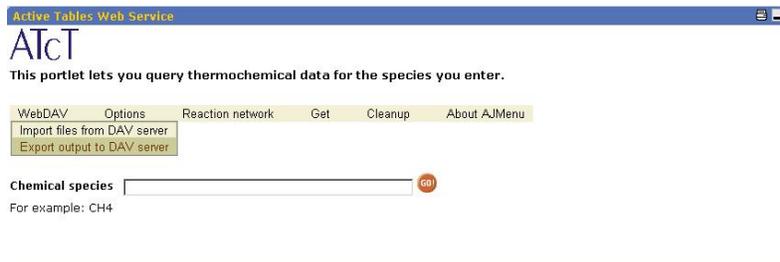


Figure 6.3: Menu — Import from DAV



Figure 6.4: Import from DAV

- **Export output to DAV Server:** This command is used to export the output from the 'solve' command to the DAV server, so that they can be further shared with others. Currently only the results of a successful query can be exported to the DAV server, and that is the file that is mentioned in the text box.

On selecting this option, shown in 6.6 the portlet displays the a message stating that you can specify the location to export the file to in the 'WebDAV URL' text box, as shown in Figure 6.7.

Clicking the 'Export result' button will then instruct the ATcT portlet to export the file from the ATcT kernel staging area to the DAV server, as shown in Figure 6.7 and Figure 6.8.

- **Options:** This menu item is used to set the following modes of functioning of the ATcT kernel.
  - **Search Mode:** This sub-menu item is used to set the following search mode options:
    - \* **Desperate Search:** has the following options:



Figure 6.5: File imported

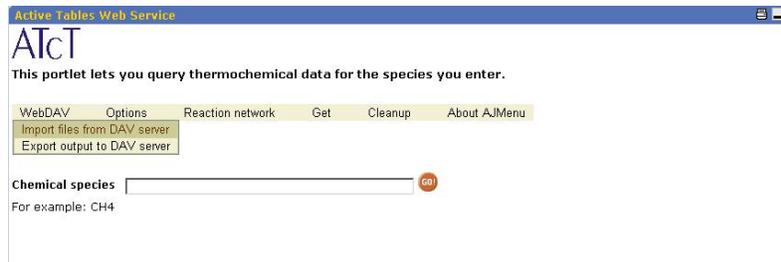


Figure 6.6: Menu — Export to DAV



Figure 6.7: Specifying the, Export-to, DAV URL

- On
- Off
- Default

Setting the Desperate search mode to on is shown in Figure 6.9 and 6.10.

\* **Hierarchical Search:** has the following options:

- On
- Off
- Default

– **Temperature Schedule:** This sub-menu item is used to set the temperature schedule for the next set of commands to the ATcT kernel.

\* **ATcT:** the ATcT temperature schedule is selected.



Figure 6.8: File exported

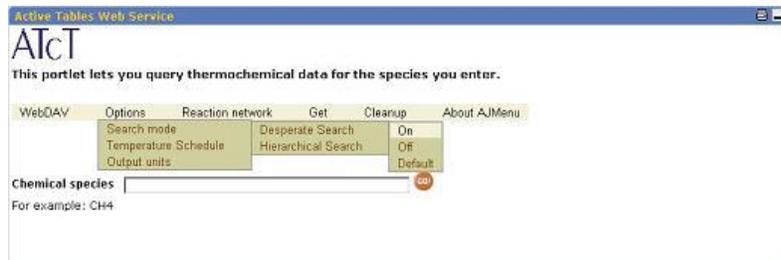


Figure 6.9: Menu — Options — Search Mode — Desperate Search — On



Figure 6.10: Desperate search mode turned on

- \* **ATcT Big**: the ATcT Big temperature schedule is selected.
- \* **Chemkin**: the Chemkin temperature schedule is selected.
- \* **Gurvich**: the Gurvich temperature schedule is selected.
- \* **Gurvich Big**: the Gurvich Big temperature schedule is selected.
- \* **JANAF**: the JANAF temperature schedule is selected.

Setting a particular temperature schedule and its output, eg: setting for ATcT Big as shown in Figure 6.11 results in the setting of the temperature schedule, as shown in Figure 6.12.

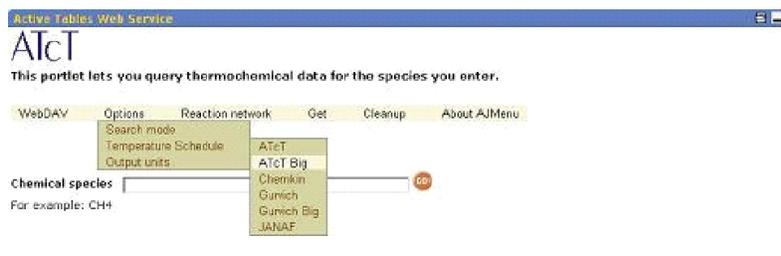


Figure 6.11: Menu — Options — Temperature Schedule — ATcT Big

- **Output Units**: This sub-menu item is used to set the output units of the query and solve operations.
  - \* **J**: sets the units to Joules.
  - \* **Cal**: sets the units to Calories.

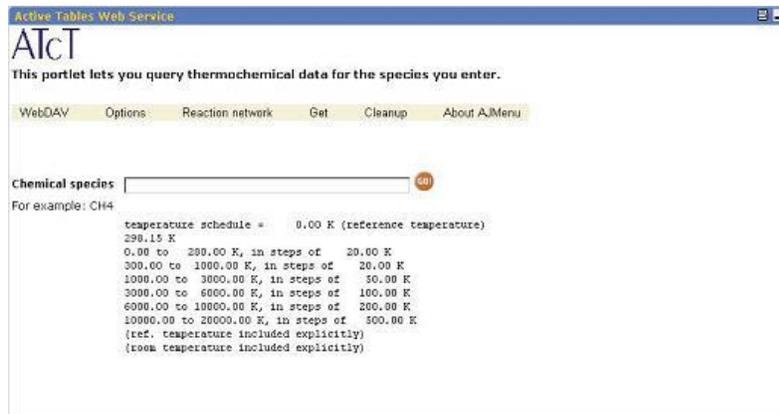


Figure 6.12: Temperature schedule set to ATcT Big

\* **R**: sets the units to R.

Setting the output units to a particular type, eg: setting the output units to Cal as shown in Figure 6.13 results in setting the output units, as shown in Figure 6.14.

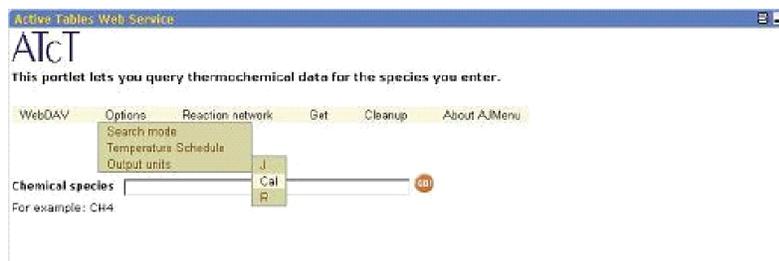


Figure 6.13: Menu — Options — Output Units — Cal



Figure 6.14: Output units set to Calories

- **Reaction Network**: This menu item is used to work with the Reaction Network imported to the ATcT kernel staging area.
  - **Solve**: This sub-menu item is used to instruct the ATcT kernel to solve the currently staged reaction network, as shown in Figure 6.15, and the results of the solve are shown in Figure 6.16.

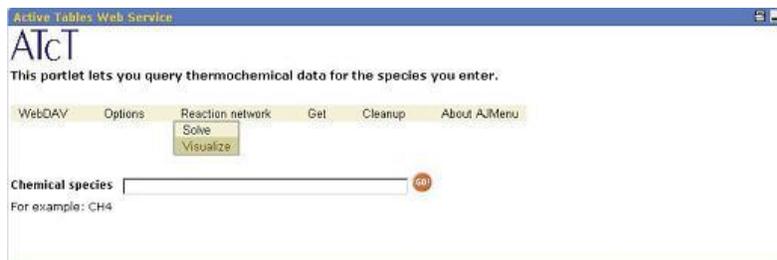


Figure 6.15: Menu — Reaction Network — Solve

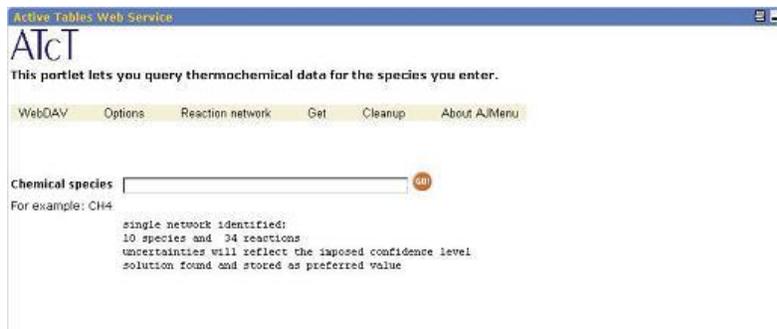


Figure 6.16: Results of the Solve

- **Visualize:** This sub-menu item is used to visualize the currently solved network, and this is presented in the Data Area of the ATcT portlet, as shown in Figure 6.17.

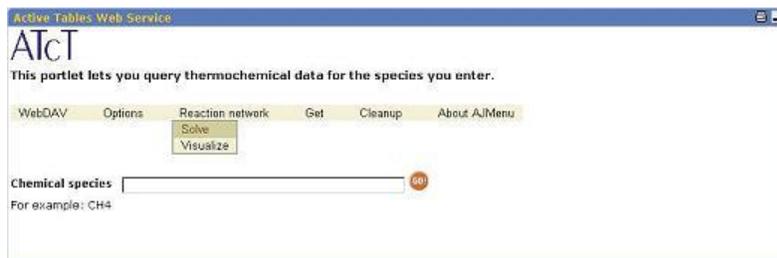


Figure 6.17: Menu — Reaction Network — Visualize

The result of the visualize command is as shown in Figure 6.18, and the image can also be zoomed in and out of.

The resultant image is clickable for each of the reactions, for which it displays, in another window, the reaction with the latest annotated simultaneous solution to the thermochemical network. For example, in the previous graph, clicking on number 14 gives us Figure 6.19.

- **Get:** menu item is used to query for the following:
  - **Search Mode:** This sub-menu item can be used to query for the modes set for the following search types:
    - \* **Desperate Search:**

\* **Hierarchical Search:**

- **Temperature Schedule:** This sub-menu item displays the selected temperature schedule.
- **Output Units:** This sub-menu item displays the currently set output units being used the the ATcT kernel.

For example, querying on the Temperature schedule, as shown in Figure 6.20, gives the output as shown in Figure 6.21.

- **Cleanup:** Use this menu item with care; because it permanently deletes all the work that is currently staged in the user working area on the ATcT kernel. The output would be as shown in Figure 6.22.

## Using the Command Box

The Command Box can be used to input the species to be queried for, such as 'CH4' or 'H2O', or others. For example querying on CH4, as shown in Figure 6.23 gives the output as shown in Figure 6.24.

Other commands can also be given through the Command Box, for example you can set the library sequence that is searched for through for the species for their enthalpy values when solving a network, through the use of the command, such as:

```
set sequence <library/notes-name>
```

For example setting the library search sequence to 'user', as shown in Figure 6.25, gives us the response as shown in Figure 6.26. A query for 'CH4' gives a result, shown in Figure 6.27, which is different as compared to Figure 6.24.

### 6.1.2 Active Tables Web (Portlet) Client for Jetspeed

Please see sub-section 6.1.1 for more details.

## 6.2 Active Tables Java Swing Client

The usability of the Active Tables Java Swing Client is envisioned to be quite different, partly because of the environment that it works in (ie, more can be done through Java Swing, as opposed to portlets), and taking into consideration the enhancements planned on the Active Tables Web Service. The major changes envisioned are:

- Explicit login/logoff mechanism, for Administrators, Registered Users and Guests.
- Browser like “browsing” of the history of the queries made on species, and the results.
- Capability of viewing/browsing, in the File Explorer style, of the user objects and access on those objects.



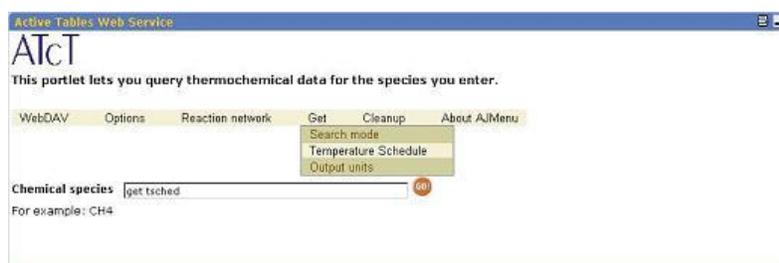


Figure 6.20: Menu — Get — Temperature Schedule

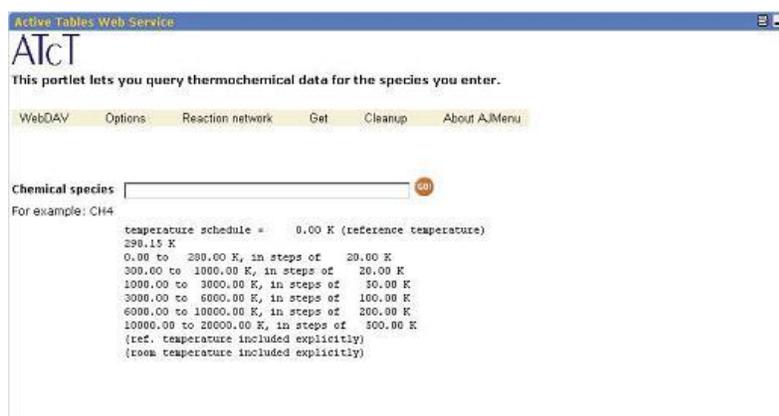


Figure 6.21: The current temperature schedule

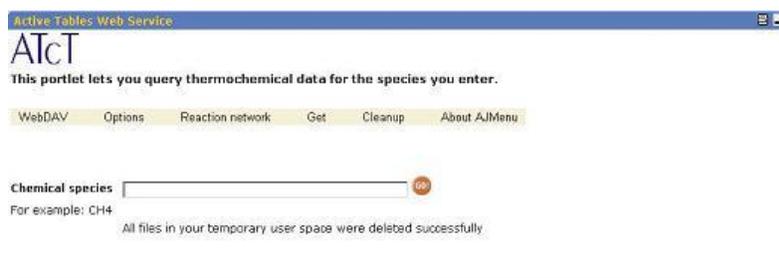


Figure 6.22: Cleaning up the working area

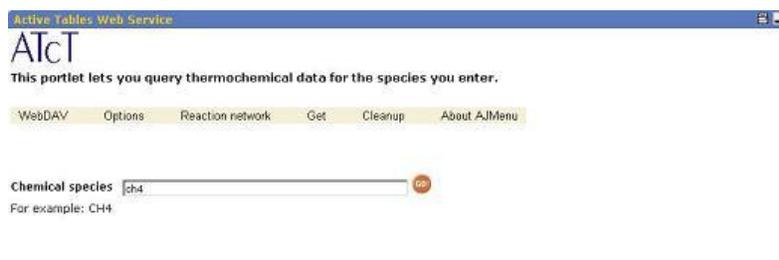


Figure 6.23: Query for CH4

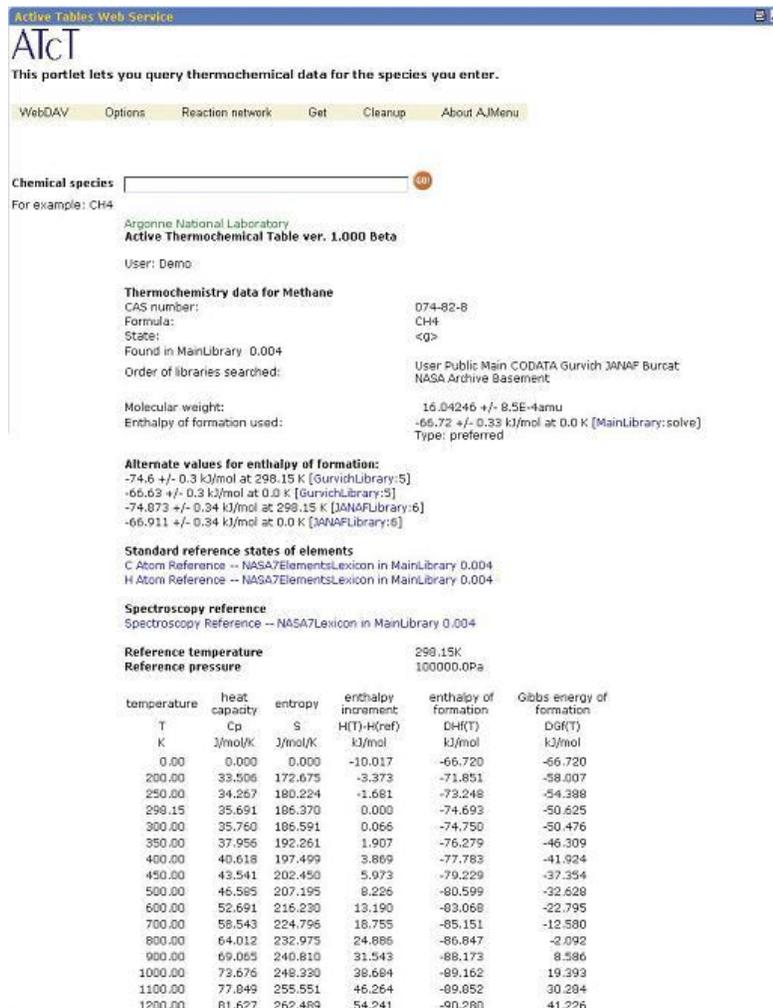


Figure 6.24: Results of the query for CH4

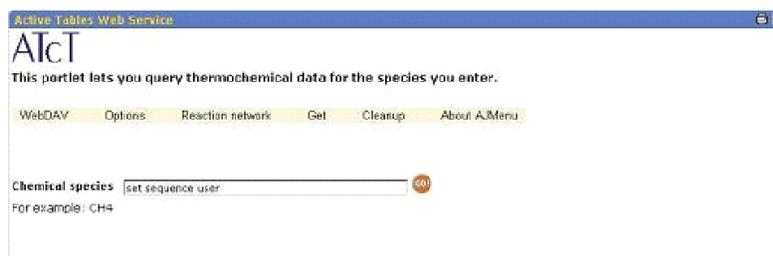


Figure 6.25: Command Box — 'set sequence user'

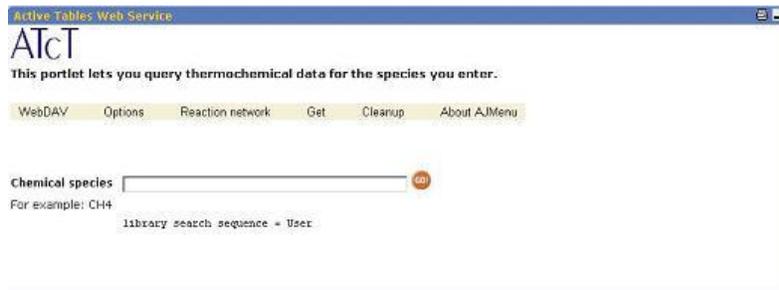


Figure 6.26: Results of 'set sequence user'

Active Tables Web Service  
**ATcT**  
 This portlet lets you query thermochemical data for the species you enter.

WebDAV Options Reaction network Get Cleanup About AJMenu

Chemical species  GO  
 For example: CH4

Argonne National Laboratory  
**Active Thermochemical Table ver. 1.000 Beta**

User: shankar

**Thermochemistry data for Methane**

CAS number: 74-82-8  
 Formula: CH4  
 State: <G>  
 Found in shankarNotes: 0.033  
 Order of libraries searched: User

Molecular weight: 16.04246 +/- 0.5E-4amu  
 Enthalpy of formation used: -17.83 +/- 0.072 kcal/mol at 298.15 K [shankarNotes:5]  
 Type: fixed

**Standard reference states of elements**  
 C Atom Reference -- NASA7ElementsLexicon in shankarNotes 0.033  
 H Atom Reference -- NASA7ElementsLexicon in shankarNotes 0.033

**Spectroscopy reference**  
 Spectroscopy Reference -- NASA7Lexicon in shankarNotes 0.033

**Reference temperature** 298.15K  
**Reference pressure** 100000.0Pa

temperature	heat capacity	entropy	enthalpy increment	enthalpy of formation	Gibbs energy of formation
T	Cp	S	H(T)-H(ref)	DH(T)	DG(T)
K	cal/mol/K	cal/mol/K	kcal/mol	kcal/mol	kcal/mol
0.00	0.000	0.000	0.000	-15.924	-15.924
200.00	8.008	41.270	1.588	-17.150	-13.842
220.00	8.067	42.036	1.748	-17.280	-13.505
240.00	8.142	42.741	1.911	-17.415	-13.156
260.00	8.246	43.397	2.074	-17.555	-12.795
280.00	8.381	44.013	2.241	-17.698	-12.424
298.15	8.530	44.544	2.394	-17.830	-12.077
300.00	8.547	44.596	2.410	-17.843	-12.042
320.00	8.740	45.154	2.583	-17.990	-11.650
340.00	8.956	45.690	2.760	-18.136	-11.249
360.00	9.192	46.209	2.941	-18.282	-10.840
380.00	9.443	46.712	3.127	-18.426	-10.423
400.00	9.708	47.203	3.319	-18.568	-9.998
420.00	9.982	47.684	3.516	-18.709	-9.566
440.00	10.264	48.155	3.718	-18.846	-9.127
460.00	10.551	48.617	3.926	-18.981	-8.683
480.00	10.841	49.072	4.140	-19.113	-8.232

Figure 6.27: Results of the query for CH4

---

## 7 Appendix

---

### 7.1 FAQs

NOTE: This should be a continuously updated section of the manual.

This section comprises the Frequently Asked Questions (FAQs) about the Active Tables application and its releases, and the answers to those questions.

- What is this FAQ for?

Answer This FAQ answers some of the common questions raised, when working with the Active Tables application.

- 

Answer

- Security:

1. Why isn't standard jstpeed (or whatever) stuff being used for user management?

Answer There are two issues here:

- (a) Authentication: The Qengine kernel has its own mechanism of security; which works on cleartext passwords. But as the service runs behind a firewall, through a Web Service interface, the cleartext passwords are not that critical.

The web-service runs within the axis web-application and it does not have support for users and management thereof.

One option is to design the wrapper to try to provide for the same. The other option is to the fortran code to implement a md5 or sha1 on the passwords stored in the txt files; Or maybe move to another form of authentication.

- (b) Authorization: At the moment the authorization scheme is based on cleartext based on a username/password scheme. Also this is sent over a non-encrypted connection; which is also not secure.

The design of the new Web Service, provides for better forms of authentication with username/credential pairs, and the use of security tokens, with HTTPS being used for encryption.

- 2.

Answer

- Active Tables Web Service:

1. Why do I get a C:\WEB-INF\lib not found. when I run ant in the web-service directory.

Answer Please check whether the `servlet.lib.dir`, and the `deploy.dir` have been specified in the `web-service/project.properties`.

2.

Answer

- Active Tables Client:
  1. Why do I get a null when I try to query for a species or run any command, in the Active Tables Web (Portlet) Clients.

Answer

2.

Answer

## 7.2 Achievements

- by **Sandeep Nijsure**
  - Created a distributed architecture to interface Active Thermochemical Tables (ATcT) Fortran application with Collaboratory for Multi-scale Chemical Sciences (CMCS) infrastructure.
  - A portlet (Web Application Component) was developed to enable a user to interact with the ATcT application. This portlet is a part of the CMCS portal.
  - A web service was created to allow interaction with Fortran ATcT application in a platform and language independent manner.
  - Another web service was created to allow the users to visualize the reaction networks using ATT GraphViz application and SVG visualizer.
  - The portlet interacts with these two web services, and hence with the ATcT application.
  - Users can query the thermochemical data using the portlet and can visualize the thermochemistry network. They can view the metadata about various reactions in the network by clicking on a reaction in the image produced by the visualization service. They can view this image in various sizes using the zooming facility provided.
  - Users can also change different ATcT parameters using an easy-to-use menu on the portlet.
  - The portlet lets the user transfer relevant data files between a WebDAV server maintained by CMCS and a temporary user area on the server hosting the ATcT application.
  - XML schemas for the data files were created. This allows data generated by ATcT to be consumed by other chemistry applications (using XSLT translations if required) and vice-versa.
  - Conversions between XML and ATcT native format were performed transparent to the user using XSLT conversions.

- Based on the data already present with ATcT, data input by the user, and the reaction network provided by the user, ATcT calculates thermochemical data about different chemical species.
- by **Shashank Shankar**
  - Designed the new Active Tables Web Service while retaining backward's compatibility with the older versions of the Clients.
  - Implemented the Vanilla-Jetspeed portlet client for the Active Tables Web Service.
  - Installed the Active Tables Web Service, as a Windows 2000 service on bubbles.mcs.anl.gov, for maintainability and ease of administration.
  - Improved the session handling within the Active Tables application.
  - Implemented upload and import of (non-and) xml files.
  - Fixed several bugs in the Active Tables CMCS portlet and the Active Tables Web Service.
    1. Fixed the browser visualization errors.
    2. Improved the image zooming interface.
    3. Implemented IE vs Netscape compatibility.
    4. AJMenu library fix and upgrade.
    5. Fixed the, XML translation, ElementCount parser bug.
  - Re-factored, commented (with licensing information), and cleaned up the code.
  - Wrote the Active Tables manual.
  - Implemented the prototype version of the Active Tables Java Swing client.

## 7.3 TODO

This section lists out some of the TODOs for the Active Tables application.

### 7.3.1 XML & Schema

1. There may be some work to be done in all translations in general to cover the variations in `¡cmcs:speciescas¿` and `¡cmcs:speciesformula¿`. They currently assume that these guys will always have `¡rdf:bag¿s` in there. This is not true. They also allow text. (text works with demo cases, will it work with production, and also baselining of schema)
2. We may want to have a XML validation step on the Jetspeed side. We are having problems due to the non-existence of validation. For example, check the two test cases for visualization, given below.

This follows from baselining the schema; talk to David Leahy (dgleahy@sandia.gov), and Branko Ruscic (ruscic@anl.gov).

3. Exporting the result of Solve.

This has to do with definition of schema for preferredEnthalpy, and the exporting of the same schema defined by David Leahy (djleahy@sandia.gov), with Qengine kernel by Branko Ruscic (ruscic@anl.gov) modifying output.

4. Say goodbye to the temporary schema for rdf, xlink, dc, dcterms etc. and replace them with the actual schema. This may or may not be necessary.

Discuss with David Leahy (djleahy@sandia.gov).

5. The Qengine kernel's output involves the states for species. They are enclosed in "i". When the output is translated to HTML for display in the portlet, those things are interpreted as HTML tags, and ignored. E.g. i. These are needed in the output.

(DONE; using the xmp tag; will this work for all version of browsers and in the future?)

6. XSD/activeTablesData/\*.xsd: schema dependencies; There was a cyclic dependency between rdf.xsd (referring to cmcs:href), and with cmcs.xsd referring to dc.xsd, and dc.xsd referring to rdf.xsd (). So to fix this for Castor, the cmcs:href part was commented in both rdf.xsd and cmcs.xsd, and introduced into rdf.xsd.

7. The output from QEngine.exe, for thermochemistry tables produced, sometimes have out-of-range in them. Right now these values are ignored (and the for the output the corresponding line is not output), while translating from txt to xml, and hence from xml to html.

Need to discuss with David Leahy (djleahy@sandia.gov) in this regards and find a better (sophisticated) way to handle this.

8. xs:any's to be removed from schemas (Pedigree is not used by ATcT, but it is used by the output translation, and thus a schema exists)

9. SpeciesDictionary.xsd is outdated, but the translation is still valid.

10. Schemas need to be standardized and published somewhere.

Talk to David Leahy (djleahy@sandia.gov), Branko Ruscic (ruscic@anl.gov), and Sandra Bittner (bittner@mcs.anl.gov) about this.

11. dcterms:created should have which time zone? Currently GMT is being used.

Talk to David Leahy (djleahy@sandia.gov).

12. XML output is not being validated on the schemas. (partially because of the rdf:href definition including the xlink:type—role—title—href complexType, for which Castor generates code of type java.lang.string, and not object references to xlink. (and similarly the xslt for QueryOutput.xsl will have to be changed)

13. What about the case when the species exists, but there is not data in the databases, then our transform fails.

14. pedigree;dcterms:reference seems to be useless, what to do about it?

Talk to David Leahy (djleahy@sandia.gov).

15. Can FixedEnthalpiesComendium have more than one value for the Enthalpy-Value? If so, then the schema will have to be updated!

Talk to Branko Ruscic (ruscic@anl.gov).

16. Fixed/Preferred/Alternate Enthalpies of Compendium; There can be a fixed, or a preferred, and/or (multiple) alternate Enthalpies of Compendium, and looks like:

- (a) ((fixed)- (preferred)-) - (alternate)\* - [- : [0..1]]
- (b) Can there be both Fixed and Preferred existing at the same time?
- (c) Can there be more than one Preferred or Fixed?
- (d) Can Preferred / Fixed be not chosen, and Alternate be used?

Talk to David Leahy (dgleahy@sandia.gov), and Branko Ruscic (ruscic@anl.gov) about this.

17. Need to handle the case when the set sequence User command is executed, and after that a species is queried for, in the absence of SpeciesCookbook, and ElementsCookbook files. (In other cases the elements and species are resolved from other files). [solving the system, and then querying the species provides additional information in the thermochemical tables produced.]

- (a) Standard:

```
Order of libraries searched:
User Public Main CODATA Gurvich JANAF
Burcat NASA Archive Basement
```

```
Molecular weight: 16.04246 +/- 8.5E-4amu
```

```
Enthalpy of formation used:
-66.72 +/- 0.33 kJ/mol at 0.0 K
[MainLibrary:solve]
Type: preferred
```

```
Alternate values for enthalpy of formation:
-74.6 +/- 0.3 kJ/mol at 298.15 K [GurvichLibrary:5]
-66.63 +/- 0.3 kJ/mol at 0.0 K [GurvichLibrary:5]
-74.873 +/- 0.34 kJ/mol at 298.15 K [JANAFLibrary:6]
-66.911 +/- 0.34 kJ/mol at 0.0 K [JANAFLibrary:6]
```

```
Standard reference states of elements
C Atom Reference ---
NASA7ElementsLexicon in MainLibrary 0.004
H Atom Reference --
NASA7ElementsLexicon in MainLibrary 0.004
```

- (b) With set sequence User, and without SpeciesCookbook and ElementsCookbook: (this needs to be handled)

```
library search sequence = User
User ==> User Notes = shankarNotes (owner = shankar)
```

```
*****
*
* ARGONNE NATIONAL LABORATORY *
*
* Active Thermochemical Table *
*
*****
```

ATcT ver. 1.000 Beta  
Used by: shankar  
16:02:35 (UTC-05) on 2003-06-27

74-82-8  
CH4  
Methane

Molecular weight: 16.04246 +- 0.00085

Preferred enthalpy of formation (fixed):  
-74.600 +- 0.300 kJ/mol at  
298.150 K [shankarNotes:5]

Preferred enthalpy of formation will be used

Standard reference states of elements:  
C definition not found  
H definition not found

With set sequence User, and  
with SpeciesCookbook and ElementsCookbook:  
Order of libraries searched: User

Molecular weight: 16.04246 +/- 8.5E-4amu

Enthalpy of formation used:  
-74.6 +/- 0.3 kJ/mol at 298.15 K [shankarNotes:5]  
Type: fixed

Files in the Active Tables Workspace:  
<http://cmcs.ca.sandia....NASA7ElementsLexicon.xml>  
<http://cmcs.ca.sandia....FixedEnthalpiesCompendium.xml>  
<http://cmcs.ca.sandia....SpeciesDictionary.xml>  
<http://cmcs.ca.sandia....AtomicLexicon.xml>  
<http://cmcs.ca.sandia....SpeciesCookbook.xml>  
<http://cmcs.ca.sandia....ElementsCookbook.xml>  
<http://cmcs.ca.sandia....NetworkEncyclopedia.xml>  
<http://cmcs.ca.sandia....NASA7Lexicon.xml>  
<http://cmcs.ca.sandia....PolyatomicRRHOLexicon.xml>

Standard reference states of elements  
C Atom Reference --

```
NASA7ElementsLexicon in shankarNotes
H Atom Reference --
NASA7ElementsLexicon in shankarNotes
```

(c)

### 7.3.2 Security

1. Validate the portlet textbox input for some maximum length. Also, allow only letters and numbers. This is to avoid common web application attacks.
2. Access to ATcT WS on bubbles.mcs.anl.gov to be restricted from outside the MCS domain, to machines from where CMCS is going to be production/dev/test:
  - (a) Either through via firewall, or
  - (b) Or via hosts, on bubbles.mcs.anl.gov.
  - (c) Both are through MCS-SYSTEMS.
3. Admin-page/WS vs Dynamic user creation as part of QEngine, and getting of user data from DAV-space.
4. Filter all input from the command box, in the portlet, such that a user is not able to bypass security.

### 7.3.3 Other

1. Check the TODOs in the Web Service and portlet code. There are a few things to be done.
2. We have had a very annoying problem in the past. The web service seems to hang sometimes. But if you press some key in the corresponding command window, it runs again. This is obviously unacceptable. I am hoping that this problem will go away now that the web service is running on top of Tomcat. Please play around with the whole thing long enough to make sure that this problem is gone. One possibility is that this is a problem with the System.out.println in Java and nothing to do with Tomcat or Axis or CMCS. Because I have observed this problem even outside of the CMCS thing. Basically after showing some messages, it hangs. Only when you press a key, it displays the rest of the messages.
3. Currently, the users are created statically on the web service side. We need to create and destroy users dynamically. When the first request from the portal comes, a new Axis session is created. We need to create a user at this time and create a Notes folder for him. This involves copying the Users.txt and Notes.txt, adding the appropriate entries and specifying these new files to the Qengine.exe . When the session is destroyed, these things need to be removed.

Since the Java wrapper writes to these files and Qengine reads from it, there can be synchronization problems. But if Qengine is started after writing the files and stopped before destroying them, then we should be good.
4. As a consequence of the above, the name of the user to the Qengine will be a session ID or something. But in anything that is visible/ exportable for the

actual user, we want the real name to be displayed. One strategy is to replace the temporary user name in any command output with the actual user name. Same with the name of the Notes folder.

5. Check if saveUser() and setPerm() is thread-safe. Because different exportHistories may bump into each other. (export filenames are stored permanently. Is this correct?)
6. The base64binary encoding thing is platform dependent. Beware if we change platforms. (how will it effect operations, and why is it being done?)
7. If Qengine crashes while satisfying a request, the Axis server goes completely nuts, and cannot do anything. HAS to be rebooted. This may not need to be fixed, if we make sure that Qengine works perfectly for all use cases. Till now we have only worked with demo cases.
8. Pressing "Enter" should work on all web forms in the portlet.
9. Adding new commands to the menu.
10. The Customize icon shouldn't be there. Instead, there should be minimize/maximize and stuff. May be use VelocityPortlet and not CustomizerVelocity or something like that.
11. Not a TODO, but just an improvement I had thought of but I am too lazy to do . The graph visualizer routine creates all the files (gif, svg etc.), zips them up in a zip file, reads the zip file into a buffer and returns the buffer. Instead of the I/O overhead of creating a zip file and then reading from it, the zipping can be done directly into a buffer, using a ByteArrayOutputStream.
12. I dont know exactly about the future plans for interaction of CMCS with CHEF. If they decide to change the portlets to CHEF teamlets in future , same may be needed for our portlet. Ask people. Not a current TODO.
13. Currently ATcT has versioning built in. Mostly we ignore it. Currently, it may show up in a few places in query output export. Remove that if we are going to ignore that.  
  
Confirm with David Leahy (djleahy@sandia.gov) and Branko Ruscic (ruscic@anl.gov) that we are going to ignore it in long run.
14. Branko Ruscic (ruscic@anl.gov) in the past has had concerns about the information displayed in the portlet in the HTML query output. I have addressed the ones I remember, but there may be more.
15. Portlet Documentation and portlet tutorial, with screenshots, help menu, etc.  
  
To be done with Branko Ruscic (ruscic@anl.gov), and David Leahy (djleahy@sandia.gov) as reviewers.
16. Investigate the possible Slide caching issue, of when a file is imported after changes, it does not give a changed file. TOP
17. What about supporting query for not only species but for a reaction (like  $H_2O_i g_i - i 2H_i g_i + O_i g_i$ )?

Talk to David Leahy (djleahy@sandia.gov).

18. The CMCS Explorer portlet should be enabled such that it populates the vector with per-item information, like whether file or directory. (ask Carina Lansing is that has been done!)

## **7.4 Change Log**

### **7.4.1 Changes from atct-1-1 to atct-1-2-alpha**

- Fixed bugs:
  - Fixed the Web Service side Qengine.exe instance not being destroyed, bug. (this is still alpha)
  - Migrated templates, and action's from cmcstemplates, and cmcs.actions to jetspeed templates and actions.
- Enhancements:
  - Implemented upload & import of (non- and) xml files.
  - Implemented standalone vanilla jetspeed version of the ATcT portlet; with query export viewable through the portal.
  - Streamlined the build system's for the Portlet(s) and the Web Service.
  - The Active Tables manual, can be found as part of this release, version 1.2-alpha.

### **7.4.2 Changes from atct-1-0 to atct-1-0**

- Fixed bugs:
  - Bug fixes to the user-interface of the ATcT portlet for integration with CHEF version 1.1.
- Enhancements:
  - Annotated the code with the globus license, and source code formatting, and commenting.
  - Wrote and appended to the developers manuals available in the Active Tables CVS.
  - Enabled the Active Tables Web Service to run as a Windows 2000 system service, on Bubbles.

---

## 8 References

---

- [1]
- [2]
- [3]
- [4]
- [5]
- [6]
- [7]
- [8]
- [9]
- [10]
- [11]
- [12]
- [13]
- [14]
- [15]
- [16]
- [17]
- [18]
- [19]
- [20]

---

## Bibliography

---

- [1] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8-9, pp. 643–662, 2001. [Online]. Available: <http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--cog-cpe-final.pdf> 58
- [2] G. von Laszewski and K. Jackson, "SciDAC: Commodity Grid Kits, Enabling Middleware for Designing Science Applications," <http://www.cogkits.org>, Sept. 2001, till Sept. 2004. 58
- [3] G. von Laszewski, B. Ruscic, P. Wagstrom, S. Krishnan, K. Amin, S. Nijssure, R. Pinzon, M. L. Morton, S. Bittner, M. Minkoff, A. Wagner, and J. C. Hewson, "A Grid Service Based Active Thermochemical Table Framework," in *Third International Workshop on Grid Computing*, ser. Lecture Notes in Computer Science, vol. 2536. Baltimore, MD: Springer, 18 Nov. 2002, pp. 25–38. [Online]. Available: <http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--cmcs.pdf> 58
- [4] G. P. Smith, D. M. Golden, M. Frenklach, N. W. Moriarty, B. Eiteneer, M. Goldenberg, C. T. Bowman, R. K. Hanson, S. Song, William, C. Gardiner, Jr., V. V. Lissianski, and Z. Qin, "Gri-Mech: Thermodynamic Data at 298 K, of the Standard Enthalpies of Formation for Radical Species," [http://www.me.berkeley.edu/gri\\_mech/data/thermo\\_table.html](http://www.me.berkeley.edu/gri_mech/data/thermo_table.html). 58
- [5] "Collaboratory for Multi-scale Chemical Sciences (CMCS)," Web Page. [Online]. Available: <http://cmcs.ca.sandia.gov/> 58
- [6] M. Frenklach, *Combustion Chemistry*. Springer-Verlag, 1984, ch. 7 Modeling, pp. 423–453. 58
- [7] B. Ruscic, J. V. Michael, P. C. Redfern, L. A. Curtiss, and K. Raghavachari, "Simultaneous Adjustment of Experimentally Based Enthalpies of Formation of CF<sub>3</sub>X, X = nil, H, Cl, Br, I, CF<sub>3</sub>, CN, and a Probe of G3 Theory," *J. Phys. Chem. A.*, vol. 102, pp. 10 889–10 899, 1998. 58
- [8] E. J. W. Jr. and M. Wiggins, "WebDAV: IETF Standard for Collaborative Authoring on the Web," *IEEE Internet Computing*, vol. 2, no. 5, p. 34, September-October 1998. 58
- [9] J. M.W. Chase, "NIST-JANAF Thermochemical Tables. Fourth Edition," *Journal of Physical and Chemical Reference*, no. Monograph 9, 1998. 58
- [10] C. M. Pancerella, L. Rahn, and C. Yang, "The Diesel Combustion Collaboratory: Combustion Researchers Collaborating over the Internet," in *Proceedings of SC99*, Portland, OR, Nov. 13-19 1999. [Online]. Available: <http://www.supercomp.org/sc99/> 58
- [11] B. Ruscic, M. Litorja, and R. L. Asher, "Ionization Energy of Methylene Revisited: Improved Values for the Enthalpy of Formation of CH<sub>2</sub> and the Bond Energy of CH<sub>3</sub> via Simultaneous Solution of the Local Thermochemical Network," *J. Phys. Chem. A.*, vol. 103, pp. 8625–8633, 1999. 58

- [12] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996. 58
- [13] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, “A Security Architecture for Computational Grids,” in *5th ACM Conference on Computer and Communications Security*. ACM Press, Nov. 2-5 1998, pp. 83–92. [Online]. Available: <ftp://ftp.globus.org/pub/globus/papers/security.pdf> 58
- [14] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, “GASS: A Data Movement and Access Service for Wide Area Computing Systems,” in *Proceedings of IOPADS’99*. Atlanta, Georgia: ACM Press, May 1999. [Online]. Available: <ftp://ftp.globus.org/pub/globus/papers/gass.pdf> 58
- [15] W. Allcock, I. Foster, and S. Tuecke, “Protocols and Services for Distributed Data-Intensive Science,” in *ACAT2000 Proceedings*, Fermi National Accelerator Laboratory. Chicago, Oct. 16-20 2000, pp. 161–163, <http://www.globus.org/research/papers/ACAT3.pdf>. 58
- [16] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch, “A National-Scale Authentication Infrastructure,” *IEEE Computer*, vol. 33, no. 12, pp. 60–66, 2000. 58
- [17] J. Novotny, S. Tuecke, and V. Welch, “An Online Credential Repository for the Grid: MyProxy,” in *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*. San Francisco: IEEE Press, Aug. 2001. [Online]. Available: <http://www.globus.org/research/papers/myproxy.pdf> 58
- [18] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, “The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration,” Open Grid Service Infrastructure WG, Global Grid Forum, 22 June 2002. [Online]. Available: <http://www.globus.org/research/papers/ogsa.pdf> 58
- [19] “Open grid services architecture homepage,” <http://www.globus.org/ogsa>, Apr. 2002. [Online]. Available: <http://www.globus.org/ogsa> 58
- [20] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, and C. Kesselman, “Grid Service Specification,” <http://www.globus.org/research/papers/gsspec.pdf>, Feb. 2002. [Online]. Available: <http://www.globus.org/research/papers/gsspec.pdf> 58