Introduction

This documentation will help you understand, install and use the SBML2SMW-CellDesigner-Plugin. This plugin allows extracting CellDesigner model information, storing this information to a Semantic MediaWiki server and context-sensitive restoring and integration of this information in a CellDesigner model. The application consists of two parts: The CellDesigner-plugin itself which directly communicates with the CellDesigner and so has access to the CellDesigner models and the TranslationServer which receives the extracted information and translates it into Semantic MediaWiki syntax and stores it there.

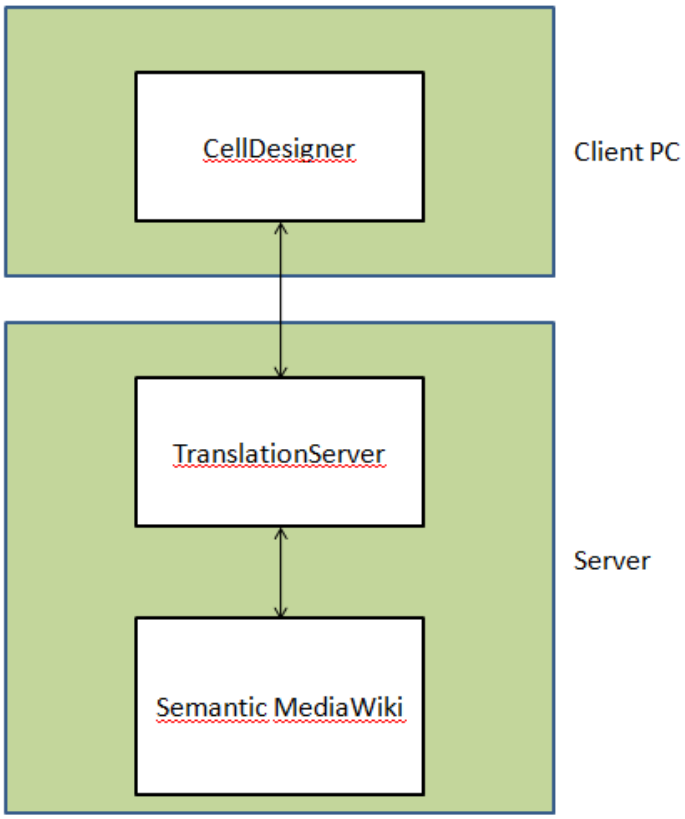


Figure 1: schematic view of the application's architecture

Figure 1 shows a schematic view of the application and its two components. Although it is possible to have the TranslationServer running on a separate system, it is recommended to have it running on the same system the Semantic MediaWiki is running in order to avoid unnecessary network traffic which would slow down the system.

Installing the application means simply installing the TranslationServer and deploying the CellDesigner plugin. After specifying the necessary configuration properties in the plugin configuration file, the plugin is directly usable. The installation itself should not take more than five minutes.

Installation

In this section, we give a very brief introduction how to get the SBML2SMW mechanisms up and running in five simple steps. Steps one to three have to be performed on every Computer which wants to use the plugin inside CellDesigner. Step four only has to be done on the servers running the Semantic MediaWiki and the TranslationServer. This does not necessarily have to be the same server, as long as all properties in SBMLconfig.prop are set correctly.

1. Copy file “SBML2SMW_plugin.jar” to the plugin-subfolder of your CellDesigner-installation-folder, this has to be done on every client machine which wants to use the plugin
2. Copy the file „SBMLconfig.prop“ to your CellDesigner- folder (i.e. where you find the file “CellDesigner4.1beta.exe”)
3. Adapt the file “SBMLconfig.prop”. It has to contain 6 lines:
    a. server=[URL of the TranslationServer]
    b. user=[MediaWiki Username with write access to the wiki]
    c. pass=[Password for MediaWiki User “user”]
    d. wikiUrl=[The URL of the MediaWiki in which the information should be stored]
    e. namespace=[The Namespace of your Semantic MediaWiki] (note: this has to be a valid RDF-Namespace, i.e. has to start with a protocol prefix like “http://”) You can check what your SMW’s namespace is in the RDF-Export. A Page named “ABC” is mapped to a URI “[namespace]/ABC” or “[namespace]/index.php/Special:URIResolver/ABC” depending on if your SMW resolves URIs or not (see 3.g)
    f. language=[the language your Semantic MediaWiki is running]
    g. resolve=[true or false] (To find out if URIResolving is activated in your SMW, go to a page containing semantic links, get the RDF-Export and check if the URIs exported contain the String “Special:URIResolver”)
4. Deploy and start the TranslationServer
    a. Copy the file “translationServer.jar” to the destination where your translationServer has to run
    b. Unzip the archive “translationServerLib.zip” to the same folder
    c. If necessary, change the port of the TranslationServer (open etc/jetty.xml, and set the default in the line starting <Set name=”port” to the port you would like to have the TranslationServer reachable, it is preset on 9999)
    d. start the server by executing startServer.cmd from a commandline
    e. Note: the server-URL in SBMLconfig.prop has to be the same as the address of the machine the TranslationServer is running, together with the port it can be reached on, e.g. http://10.212.11.13:9999
5. Start your CellDesigner, there are no more configuration steps necessary

Overview: Using the plugin

1. Start CellDesigner
2. Create a new model or open a saved model

3. Start the SBML2SMW plugin
    a. Click on “Plugin” -> “SBML2SMW” -> “Start SBML2SMW Plugin”
4. Storing information:
    a. Mark the elements you would like to store. A reaction will be stored, iff all the participating substances on this reaction are marked
    b. Click the “Store” button in the plugin window
    c. Depending on the connection between your local machine and the server, storing can take a while due to authentication and multiple write accesses to the MediaWiki

5. Loading information:
    a. Mark one or more elements of your model
    b. Click the load button in the plugin window
    c. All reactions having one of the marked elements is either as a reactant or as a product are retrieved from the MediaWiki
    d. If other reaction participants of such reactions are already present in the model, the plugin interlinks these elements, only these elements not contained in the model are added to the model

Detailed description of the plugin’s capabilities

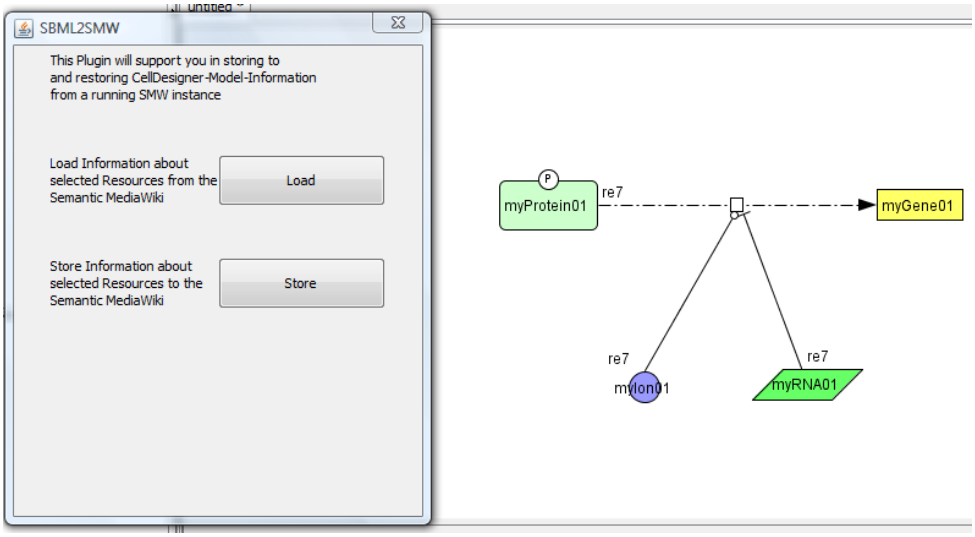
1. Stored information about species (e.g. Proteins, Genes, RNA, …)
    a. the name of the species
    b. the type of the species
    c. for proteins: the protein-type of the species (Generic, Receptor, …)
2. Stored information about reactions
    a. the name of the reaction (random name, if no name was specified in the model)
    b. the type of the reaction
    c. a list of the reactants
    d. a list of the products
    e. a list of the reaction’s modifications
3. Stored information about modifications
    a. the name of the modifier
    b. the type of the modification (catalysis, inhibition, …)
    c. a generated name for the modification

Names are primary keys for all stored elements. If user A stores information about a receptor with name XYZ, and a second user B stores information about a RNA XYZ, user B overrides the old content, the information added by user A get lost.

If a species has been marked as modified (e.g. phosphorylated), the modification is encoded in the name of the stored species. This is a result of the primary key nature of the name of species. A phosphorylated protein with name “prot” is stored under the name ”prot_modified_phosphorylated”. If this reaction is loaded with the plugin, the new model element will also carry this name

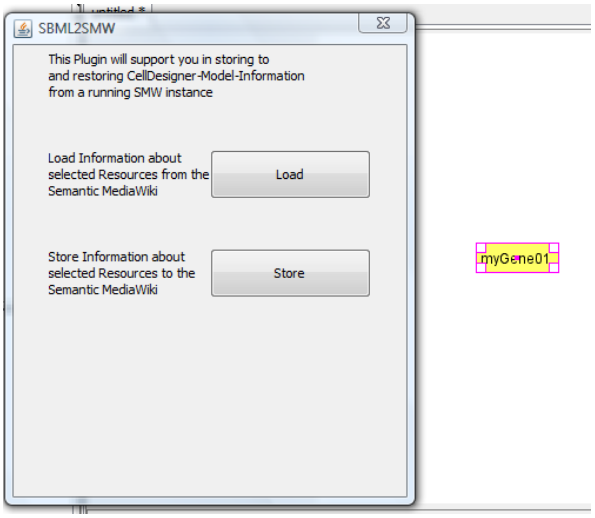
Examples

Storing a reaction

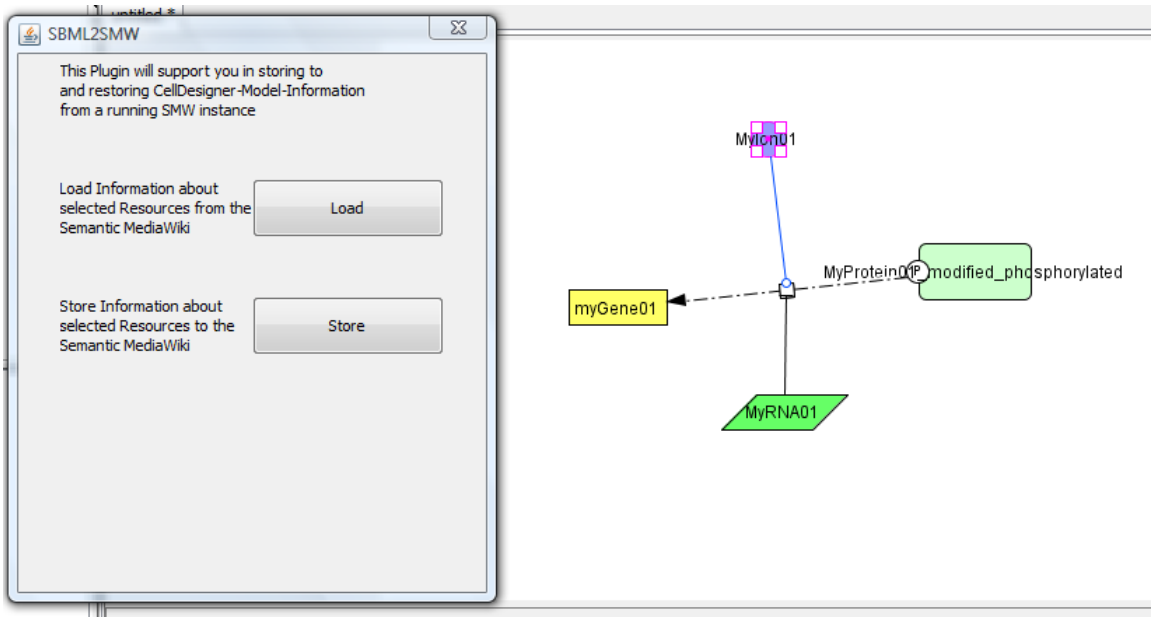


To store the reaction shown on the right, mark all four elements and click “Store”

Load a reaction

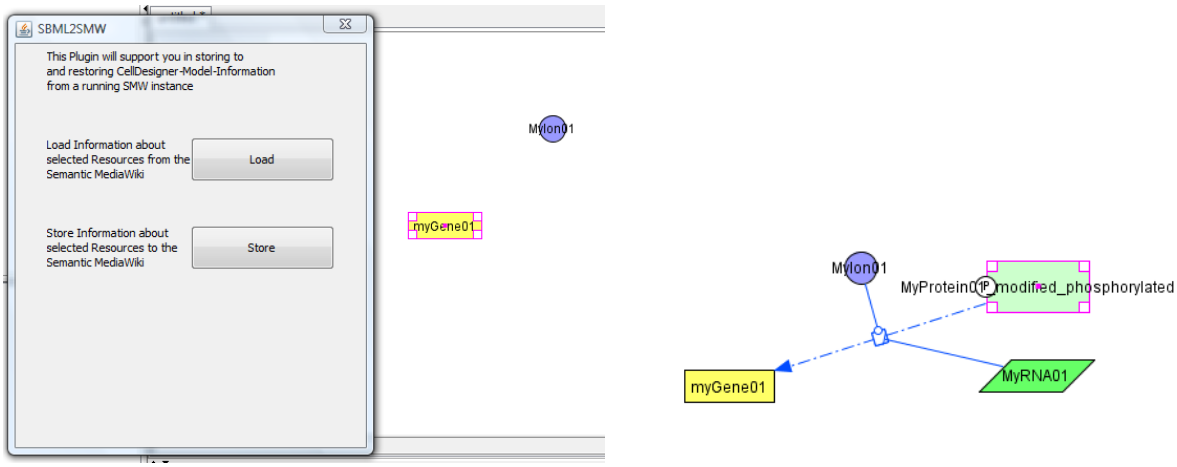


To load the reactions for an element in the model from the MediaWiki, select the element “myGene01” and click Load



The three missing elements (MyIon01, myRNA01 and MyProtein01(with its modification)) are added to the model, the reaction links them.

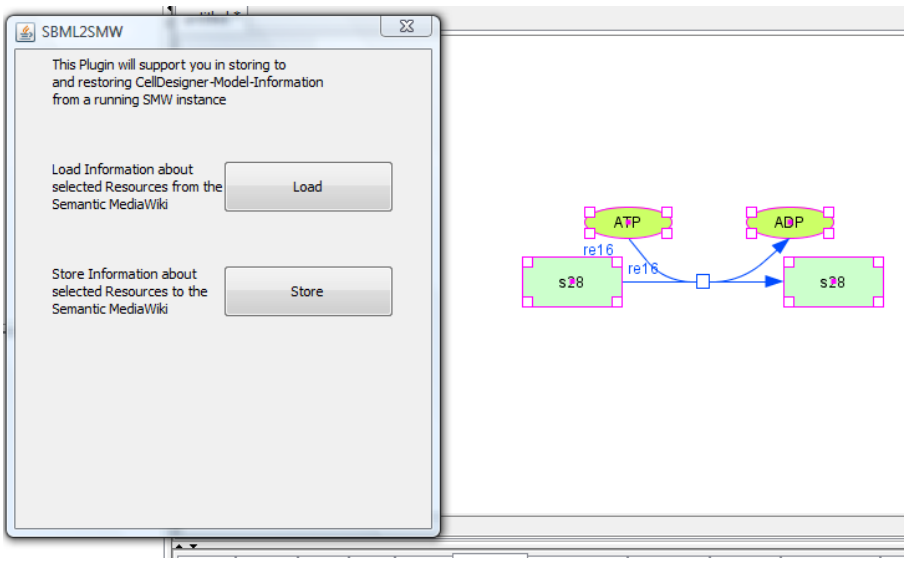
Load a reaction (some participants already in the model)

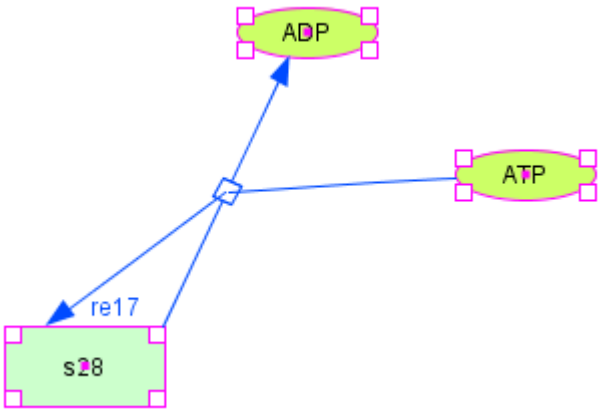


If one or more of the reaction participants is already present in the model, it is not re-added but linked to the loaded reaction. This way, the user can find relations between elements in his model.

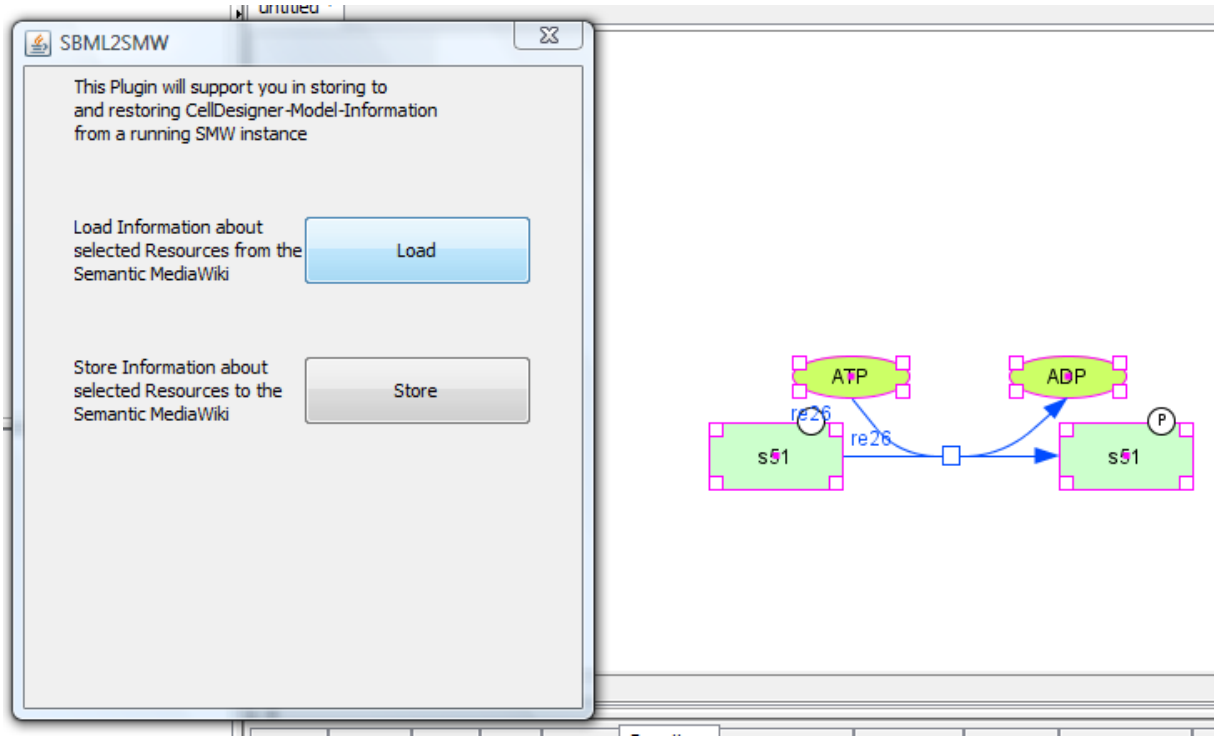
State Transitions

If participating in a state transition, the product has to in another state as the reactant, e.g. by adding a modification

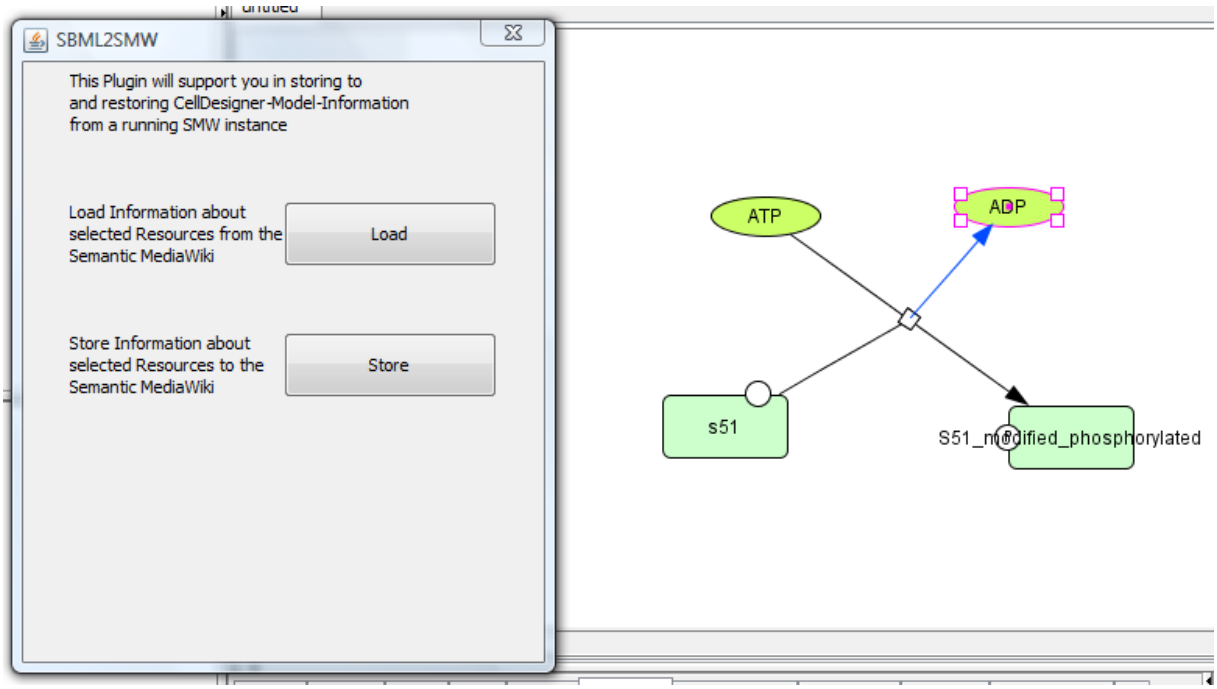




If this is not the case, CellDesigner cannot disambiguate between the reactant and the product, and links both of the edges to the same element.



If the protein in fact changed its state, the reaction is correctly loaded



Requirements

The plugin was designed for CellDesigner 4.1beta. It was tested on a system running MediaWiki 1.15.1 with MySQL 5.0.26-Max, PHP 5.1.2 and the Semantic MediaWiki extension 1.5.1.1. It currently supports english and german versions of Semantic MediaWiki. If the MediaWiki is configured to another language, according adaptions have to be made to the class Languages.LanguageSpecs in the TranslationServer.