

Planning.Domains

Christian Muise

MIT CSAIL, USA
cjmuise@mit.edu

Tutorial Overview

(5min) Introduction to Planning.Domains

(15min) `api.planning.domains`

(10min) `solver.planning.domains`

(15min) `editor.planning.domains`

(5min) *Break*

(40min) Putting it all together

- Three initiatives; one platform.
- Strong focus on planning problems.
- Tools for (and by) the community.

- Three initiatives; one platform.
- Strong focus on planning problems.
- Tools for (and by) the community.

Thanks! The planning.domains services are graciously supported by the ICAPS organization and community.

Wouldn't it be cool if...

...we had a central repository for PDDL files.

- Open repository of over 125 domains

The screenshot shows the GitHub interface for the repository `pddl-tools` under the user `domains`. The repository is categorized under `ACTIONS`. The `NAVIGATION` sidebar on the left includes links for `Overview`, `Source` (which is highlighted), `Commits`, `Branches`, `Pull requests`, `Issues` (with a badge showing 2), `Wiki`, `Downloads`, and `Settings`. The `Source` tab on the right displays a file tree for the `domains / classical /` directory. The file list includes: `..`, `airport`, `airport-adl`, `assembly`, `barman-opt11-strips`, `barman-opt14-strips`, `barman-sat11-strips`, `barman-sat14-strips`, `blocks`, `blocks-3op`, `briefcaseworld`, `cavediving`, `childsnack-opt14-strips`, `childsnack-sat14-strips`, `citycar-opt14-adl`, `citycar-sat14-adl`, and `cybersec`.

Source

master | domains / classical /

- ..
- airport
- airport-adl
- assembly
- barman-opt11-strips
- barman-opt14-strips
- barman-sat11-strips
- barman-sat14-strips
- blocks
- blocks-3op
- briefcaseworld
- cavediving
- childsnack-opt14-strips
- childsnack-sat14-strips
- citycar-opt14-adl
- citycar-sat14-adl
- cybersec

- Open repository of over 125 domains
- API access for:
 - Collections
 - Domains
 - Problems

```
GET api.planning.domains/collections
```

Returns all of the collections.

```
GET api.planning.domains/collection/{col-id}
```

Returns the collection matching `col-id`.

```
GET api.planning.domains/domain/{dom-id}
```

Returns the domain matching `dom-id`.

```
GET api.planning.domains/problems/search
```

Returns all of the problems matching the query provided. The following parameters can be used for the query:

Param	Value	Description
<code>domain</code>	Number	Matches the provided domain ID.
<code>domain_name</code>	String	Matches when the problem's domain name contains the provided string.
<code>problem_name</code>	String	Matches when the problem's name contains the provided string.
<code>min_lower_bound</code>	Number	Matches all problems with a lower bound no smaller than the provided number.
<code>max_upper_bound</code>	Number	Matches all problems with an upper bound no larger than the provided number.

- Open repository of over 125 domains
- API access for:
 - Collections
 - Domains
 - Problems
- Python library and command utility

```
> ./planning.domains.py

No command-line options given.  Usage:

planning.domains.py update                Update the local domain database

planning.domains.py find collections [string] Find collections whose names match [string]
planning.domains.py find domains [string] Find domains whose names match [string]
planning.domains.py find problems [string] Find problems whose names match [string]

planning.domains.py show collection [integer] Find collections whose id is [integer]
planning.domains.py show domain [integer] Find domains whose id is [integer]
planning.domains.py show problem [integer] Find problems whose id is [integer]
```

```
import sys

print "Loading domains...",
sys.stdout.flush()

import planning_domains_api as api

# 12 is the collection for all STRIPS IPC domains
domains = {}
for dom in api.get_domains(12):

    # Turn the links into relative paths for this machine
    probs = map(api.localize, api.get_problems(dom['id']))

    # Map the domain name to the list of domain-problem pairs
    domains[dom['dom_name']] = []
    for p in probs:
        domains[dom['dom_name']].append((p['dom_url'], p['prob_url']))

print "done!"
```


- Open repository of over 125 domains
- API access for:
 - Collections
 - Domains
 - Problems
- Python library and command utility
- JavaScript library

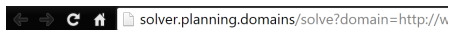
```
<script type="text/javascript" src="planning-domains.js"></script>
<script type="text/javascript">
  fetch_domains('/domains/6', '#domains', 'alert');
</script>
```

ID	Domain	Requirements	Description
80	cybersec	:action-costs :strips	A domain that models the cyber security model of vulnerability analysis for cyber defense.
58	elevators	:action-costs :typing	(opto8) The scenario is the following: There is a building with $N+1$ floors, numbered from 0 to N . The building can be separated in blocks of size $M+1$, where M divides N . Adjacent blocks have a common floor. For example, suppose $N=12$ and $M=4$, then we have 15 floors in total (ranging from 0 to 12), which form 3 blocks of 5 floors each, being 0 to 4, 4 to 8 and 8 to 12. The building has K fast (accelerating) elevators that stop only in floors that are multiple of $M/2$ (so M has to be an even number). Each fast elevator has a capacity of X persons. Furthermore, within each block, there are L slow elevators, that stop at every floor of the block. Each slow elevator has a capacity of Y persons (usually Y
95	elevators	:action-costs :typing	(sato8) The scenario is the following: There is a building with $N+1$ floors, numbered from 0 to N . The building can be separated

Wouldn't it be cool if...

...we had a planner in the cloud.

- Call via a URL
- Call using JSON

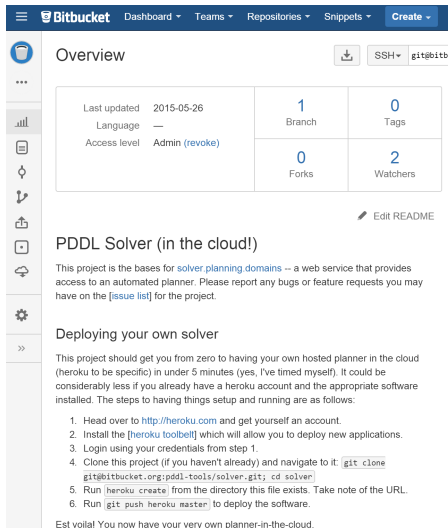


Plan Found:

```
(unstack b c)
(put-down b)
(unstack c a)
(put-down c)
(unstack a d)
(stack a b)
(pick-up c)
(stack c a)
(pick-up d)
(stack d c)
```

```
$.ajax( {url: "http://solver.planning.domains/solve",
  type: "POST",
  contentType: 'application/json',
  data: JSON.stringify({"domain": domText,
    "problem": probText})})
  .done(function (res) {
    if (res['result'] === 'ok') {
      window.alert('Plan found!');
    } else {
      window.alert('Planning failed.');
```

- Call via a URL
- Call using JSON
- FOSS project to deploy your own



The screenshot shows the Bitbucket web interface for a repository named 'PDDL Solver (in the cloud!)'. The repository is owned by 'git@bitb' and is in the 'Admin (revoke)' access level. It was last updated on 2015-05-26. The repository has 1 branch, 0 tags, 0 forks, and 2 watchers. There is a link to 'Edit README'. The repository description states: 'This project is the bases for [solver planning domains](#) -- a web service that provides access to an automated planner. Please report any bugs or feature requests you may have on the [\[issue list\]](#) for the project.'

Deploying your own solver

This project should get you from zero to having your own hosted planner in the cloud (heroku to be specific) in under 5 minutes (yes, I've timed myself). It could be considerably less if you already have a heroku account and the appropriate software installed. The steps to having things setup and running are as follows:

1. Head over to <http://heroku.com> and get yourself an account.
2. Install the [\[heroku toolbelt\]](#) which will allow you to deploy new applications.
3. Login using your credentials from step 1.
4. Clone this project (if you haven't already) and navigate to it: `git clone git@bitbucket.org:pddl-tools/solver.git; cd solver`
5. Run `heroku create` from the directory this file exists. Take note of the URL.
6. Run `git push heroku master` to deploy the software.

Est voila! You now have your very own planner-in-the-cloud.

- Call via a URL
- Call using JSON
- FOSS project to deploy your own
- Ultra-agile track for king-of-the-hill



$\equiv BFS(f)$

Wouldn't it be cool if...

...we had a dedicated editor for PDDL.

- Online editor



Editor

- Online editor
- Syntax highlighting

```
1  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
2  ;; 4 Op-blocks world
3  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4
5  (define (domain BLOCKS)
6    (:requirements :strips)
7    (:predicates (on ?x ?y)
8      (ontable ?x)
9      (clear ?x)
10     (handempty)
11     (holding ?x)
12    )
13
14  (:action pick-up
15    :parameters (?x)
16    :precondition (and (clear ?x) (ontable ?x) (handempty))
17    :effect
18    (and (not (ontable ?x))
19      (not (clear ?x))
20      (not (handempty))
21      (holding ?x)))
22
```

```
1  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
2  ;; 4 Op-blocks world
3  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4
5  (define (domain BLOCKS)
6    (:requirements :strips)
7    (:predicates (on ?x ?y)
8      (ontable ?x)
9      (clear ?x)
10     (handempty)
11     (holding ?x)
12    )
13
14  (:action pick-up
15    :parameters (?x)
16    :precondition (and (clear ?x) (ontable ?x) (handempty))
17    :effect
18    (and (not (ontable ?x))
19      (not (clear ?x))
20      (not (handempty))
21      (holding ?x)))
22
```


- Online editor
- Syntax highlighting
- Bracket folding

```
4
5 (define (domain BLOCKS)
6   (:requirements :strips)
7   (:predicates (on ?x ?y)
8                 (ontable ?x)
9                 (clear ?x)
10                (handempty)
11                (holding ?x)
12                )
13
14 ▾ (:action pick-up
15   :parameters (?x)
16   :precondition (blue)
17   :effect (blue))
23
24 ▾ (:action put-down
25   :parameters (?x)
26   :precondition (holding ?x)
27   :effect
28   (and (not (holding ?x))
29        (clear ?x)
30        (handempty)
31        (ontable ?x)))
32 ▾ (:action stack
33   :parameters (?x ?y)
34   :precondition (and (holding ?x) (clear ?y))
35   :effect
36   (and (not (holding ?x))
37        (not (clear ?y))
38        (clear ?x)
39        (handempty)
40        (on ?x ?y)))
```

Editor

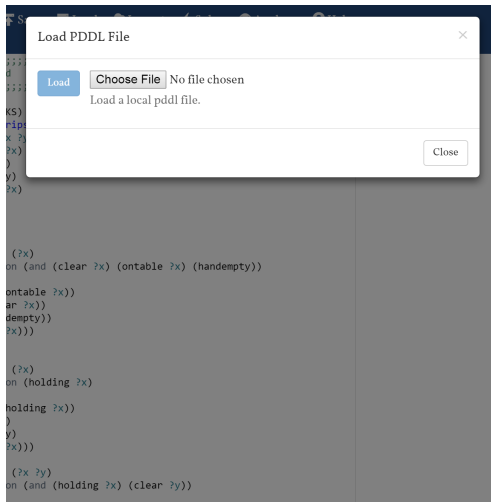
- Online editor
- Syntax highlighting
- Bracket folding
- Auto-completion

```
26      :precondition (holding ?x)
27      :effect
28      (and (not (holding ?x))
29            (clear ?x)
30            (handempty)
31            (ontable ?x)))
32  (:action stack
33    :parameters (?x ?y)
34    :precondition (and (holding ?x) (clear ?y))
35    :effect
36    (and (not (holding ?x))
37          (not (clear ?y))
38          (clear ?x)
39          (handempty)
40          (on ?x ?y)))
41  (:action unstack
42    :parameters (?x ?y)
43    :precondition (and (on ?x ?y)
44                      (and (holding ?x)
45                            (clear ?y)))
46    :effect
47    (and (clear ?x)
48          (handempty)
49          (on ?x ?y)))
50  (:action durative-action
51    :parameters (?x ?y ?z)
52    :precondition (and (holding ?x)
53                      (clear ?y)
54                      (clear ?z))
55    :effect
56    (and (clear ?x)
57          (clear ?y)
58          (clear ?z)
59          (on ?x ?y)
60          (on ?x ?z)))
61  )
62  )
63  )
64  )
65  )
66  )
67  )
68  )
69  )
70  )
71  )
72  )
73  )
74  )
75  )
76  )
77  )
78  )
79  )
80  )
81  )
82  )
83  )
84  )
85  )
86  )
87  )
88  )
89  )
90  )
91  )
92  )
93  )
94  )
95  )
96  )
97  )
98  )
99  )
100 )
```

action
(:action \${I:actionName})
:parameters (?x - type)
:precondition (and (foo ?x))
:effect (and
(fuu ?x)
(not (fuu ?x ?x))
)
)

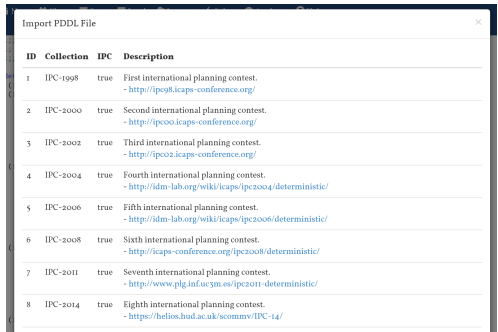
Editor

- Online editor
- Syntax highlighting
- Bracket folding
- Auto-completion
- Save/load locally



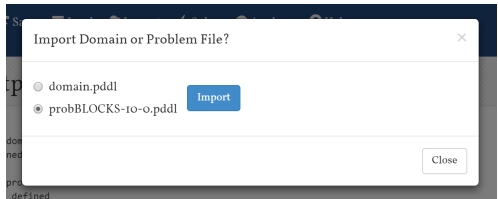
Editor

- Online editor
- Syntax highlighting
- Bracket folding
- Auto-completion
- Save/load locally
- Import via the API



Import PDDL File

ID	Collection	IPC	Description
1	IPC-1998	true	First international planning contest. - http://ipc98.icaps-conference.org/
2	IPC-2000	true	Second international planning contest. - http://ipc00.icaps-conference.org/
3	IPC-2002	true	Third international planning contest. - http://ipc02.icaps-conference.org/
4	IPC-2004	true	Fourth international planning contest. - http://idm-lab.org/wiki/caps/ipc2004/deterministic/
5	IPC-2006	true	Fifth international planning contest. - http://idm-lab.org/wiki/caps/ipc2006/deterministic/
6	IPC-2008	true	Sixth international planning contest. - http://icaps-conference.org/ipc2008/deterministic/
7	IPC-2011	true	Seventh international planning contest. - http://www.plg.inf.uc3m.es/ipc2011-deterministic/
8	IPC-2014	true	Eighth international planning contest. - https://helios.hud.ac.uk/scommv/IPC-14/



Import Domain or Problem File?

☐ domain.pddl

☒ probBLOCKS-10-o.pddl

Import

Close

Editor

- Online editor
- Syntax highlighting
- Bracket folding
- Auto-completion
- Save/load locally
- Import via the API
- Compute plans via the online solver

The image shows a 'Compute Plan' dialog box with the following fields:

- Domain:** domain.pddl
- Problem:** probBLOCKS-7-o.pdc
- Custom Planner URL:** http://solver.planning.domains

A green 'Plan' button is visible. Below the dialog, the 'Found Plan' section displays a list of actions and their corresponding Prolog code:

domain.pddl	Found Plan
probBLOCKS-7-o.pddl	(unstack e g)
Plan (t)	(put-down e)
	(unstack g b)
	(put-down g)
	(unstack b a)
	(put-down b)
	(unstack a f)
	(put-down a)
	(unstack f c)

The Prolog code for the found plan is:

```
(:action put-down
:parameters (b)
:precondition
(holding b)
:effect
(and
(not
(holding b)
)
(clear b)
(handempty)
(ontable b)
)
)
```

- Online editor
- Syntax highlighting
- Bracket folding
- Auto-completion
- Save/load locally
- Import via the API
- Compute plans via the online solver
- Analyze using TorchLight

Torchlight Output

```
TorchLight: parsing domain file
domain 'BLOCKS' defined
... done.
TorchLight: parsing problem file
problem 'BLOCKS-7-0' defined
... done.

TorchLight: running Fast-Downward translator to generate variables ... done.
TorchLight: creating SG and DTG structures
Warning: didn't find variable value for FF ft ON(E E). Skipping the fact from variables structures.
Warning: didn't find variable value for FF ft ON(G G). Skipping the fact from variables structures.
Warning: didn't find variable value for FF ft ON(B B). Skipping the fact from variables structures.
Warning: didn't find variable value for FF ft ON(A A). Skipping the fact from variables structures.
Warning: didn't find variable value for FF ft ON(F F). Skipping the fact from variables structures.
Warning: didn't find variable value for FF ft ON(C C). Skipping the fact from variables structures.
Warning: didn't find variable value for FF ft ON(D D). Skipping the fact from variables structures.
TorchLight: static examination of SG and DTG structures ... done.

TorchLight guaranteed global analysis:
Failed.
Percentage of successful x0/t0 gDGs : 7.06% (6 of 85)

TorchLight guaranteed local analysis of initial state:
Failed.

TorchLight approximate local analysis of initial state:
Failed.

TorchLight: sampling random states ... done.

TorchLight guaranteed local analysis of sampled states:
Success and hence no local minima under h+: 0.00%
```

- Online editor
- Syntax highlighting
- Bracket folding
- **Auto-completion**
- Save/load locally
- Import via the API
- **Compute plans via the online solver**
- **Analyze using TorchLight**

Anatomy of a Plugin (JavaScript file)

```
define(function () {  
  return {  
    name: "Plan-o-matic_1000",  
    author: "John_Smith",  
    email: "yeah@right.com",  
    description: "A_plugin_template.",  
  
    // Called when loaded or enabled  
    initialize: function() { },  
  
    // Called when disabled  
    disable: function() { },  
  
    // Used to save settings  
    save: function() { return {}; },  
  
    // Restore any previous settings  
    load: function(settings) { }  
  };  
});
```

planning.domains

1 **api.planning.domains**

- Central PDDL repository
- API interface to all benchmarks
- Suite of tools to interface with API

2 **solver.planning.domains**

- Planner in the cloud
- Open source project
- Rolling ultra-agile contest

3 **editor.planning.domains**

- Custom PDDL editor
- Tie-in to the API and Solver
- TorchLight and other analysis soon

- **JSON read and write API**
- Python interface
- JavaScript widgets
- Command-line utility

API: Constructing a URL

```
http://api.planning.domains
```

The start of every API call

API: Constructing a URL

```
http://api.planning.domains  
/<format>
```

Can be **json** or **xml** (ommitted for POST)

API: Constructing a URL

```
http://api.planning.domains  
    /<format>  
    /<genre>
```

For now, can only be **classical**
(*FOND, RDDDL, RMPL, etc coming soon*)

API_PATH: `api.planning.domains/json/classical`

API: Constructing a URL

```
http://api.planning.domains  
    /<format>  
    /<genre>  
    /[collection|domain|problem] (s)
```

Depends on if you want a list or single object

API: Constructing a URL

```
http://api.planning.domains  
  /<format>  
  /<genre>  
  /[collection|domain|problem] (s)  
  /<id>|search?option=val|...
```

Options vary after this point...

Collections

```
api.planning.domains/json/classical/collection/12
{
  "error": false ,
  "message": "Success!" ,
  "result": {
    "collection_id": 12,
    "collection_name": "All-IPC (STRIPS)" ,
    "description": "A selection of STRIPS..." ,
    "domain_set": "[8,14,17,19,24,27,...,129]" ,
    "tags": "[]"
  }
}
```

Domains

```
api.planning.domains/json/classical/domain/13
{
  "error": false ,
  "message": "Success!" ,
  "result": {
    "domain_id": 13,
    "domain_name": "transport" ,
    "description": "(opt11) Each vehicle..." ,
    "tags": "[\":action-costs\", \":typing\"]"
  }
}
```


Problems

api.planning.domains/json/classical/problem/13

```
{ "error": false ,
  "message": "Success!" ,
  "result": {
    "problem_id": 13,
    "domain_id": 4,
    "domain": "sokoban" ,
    "problem": "p06.pddl" ,
    "domain_url": "http://www.haz.ca/planning-domains/..." ,
    "problem_url": "http://www.haz.ca/planning-domains/..." ,
    "domain_path": "classical/sokoban-opt08-strips/p06-domain.pddl" ,
    "problem_path": "classical/sokoban-opt08-strips/p06.pddl" ,
    "tags": "[]" ,
    "lower_bound": 5,
    "upper_bound": 11,
    "average_effective_width": null ,
    "max_effective_width": null ,
    "lower_bound_description": "haslum/pd-missing-hlb/..." ,
    "upper_bound_description": "Resetting the upper bounds" ,
    "average_effective_width_description": " " ,
    "max_effective_width_description": " "
  }
}
```

Demo!

Submitting Attributes

Note: For now, modifications require special access

- Can update the attribute of any type:
 - POST **API_PATH**/updatecollection/{col-id}
 - POST **API_PATH**/updatedomain/{dom-id}
 - POST **API_PATH**/updateproblem/{prob-id}
- Following parameters are required:

user	Email address
password	Provided by admins
key	Attribute name
value	New value
desc	Description indicating source

API Null Attributes

To find all problems that have a particular attribute set to **null**. For example, all problems missing an upper bound:

GET **API_PATH**/nullattribute/upper_bound

```
{ "error": false ,
  "message": "Success!" ,
  "result": (
    {
      "id": 3027 ,
      "domain_path": "classical/pathways/domain_p02.pddl" ,
      "problem_path": "classical/pathways/p02.pddl"
    } ,
    {
      "id": 485 ,
      "domain_path": "classical/floortile-opt11-strips/domain.pddl" ,
      "problem_path": "classical/floortile-opt11-strips/opt-p09-018.pddl"
    } ,
    ...
  )
}
```

Examples: PDDL Requirements, Unsolvables, Invertables

Note: For now, modifications require special access

- Listing all tags:

- GET **API_PATH**/tags

- Adding tags:

- POST **API_PATH**/tagcollection/{col-id}
- POST **API_PATH**/tagdomain/{dom-id}
- POST **API_PATH**/tagproblem/{prob-id}

- Removing tags:

- POST **API_PATH**/untagcollection/{col-id}
- POST **API_PATH**/untagdomain/{dom-id}
- POST **API_PATH**/untagproblem/{prob-id}

Incumbent can be submitted and retrieved

- GET **API_PATH**/plan/{prob-id}

```
{ "error": false ,  
  "message": "Success!",  
  "result": {  
    "plan": "(move player-01 pos-6-4 pos-6-3 dir-up)\n..."  
  }  
}
```
- POST **API_PATH**/submitplan/{prob-id}
plan String of IPC-style plan
email User email (for the glory)

Note: No special access required!

Demo!

- **JSON API access**
- ~~Open source project~~
- ~~Ultra-agile track~~

Three main `POST` endpoints for solving and validating:

```
solver.planning.domains/solve
```

```
solver.planning.domains/validate
```

```
solver.planning.domains/solve-and-validate
```

<code>domain</code>	Either URL or raw PDDL for domain
<code>problem</code>	Either URL or raw PDDL for problem
<code>probID</code>	API ID to supersede <code>domain</code> and <code>problem</code>
<code>is_url</code>	Set to true if using URLs
<code>plan</code>	IPC format plan (just for <code>/validate</code>)

Solver API

Returned parameters if plan is computed:

length	Number of actions
output	Planner output
parse_status	Status of the plan parsing (e.g., ok)
type	Either simple or full
plan	...

Returned parameters if VAL is called:

cost	Total plan cost
val_stdout	VAL standard output
val_stderr	VAL standard error
val_status	Either valid or err
error	Indication of any VAL error

Solver API: Returned Plan

(full)

Ground action info included

Array of objects

name Ground action name

action Full ground action

```
"plan": {  
  { "action":  
    " (:action move\n...",  
    "name":  
      "(move player-01..." },  
  { "action":  
    " (:action move\n...",  
    "name":  
      "(move player-02..." },  
  ...  
}
```

(simple)

Parser was unable to ground

Array of strings

```
"plan": (  
  "(move player-01 pos-6-4...",  
  "(move player-02 pos-3-5...",  
  "(move player-01 pos-6-3...",  
  ...  
)
```

Over 14k plans computed
since announcing at ICAPS!

Demo!

- Editor Usage
- Editor Plugin Architecture

Demo!

Editor Plugin Structure

```
define(function () {  
    return {  
        name: "Plan-o-matic 1000",  
        author: "John Smith",  
        email: "yeah@right.com",  
        description: "A plugin template.",  
  
        // Called when loaded or enabled  
        initialize: function() { },  
  
        // Called when disabled  
        disable: function() { },  
  
        // Used to save settings  
        save: function() { return {}; },  
  
        // Restore any previous settings  
        load: function(settings) { }  
    };  
});
```

Editor Meta Plugin Structure

```
define(function () {  
    return {  
        // Mandatory flag  
        meta: true ,  
  
        // List of meta / normal plugins  
        plugins: {  
            "plugin1":  
                {url: "http://path.to.plugin/1",  
                 settings: {} },  
            "plugin2":  
                {url: "http://path.to.plugin/2",  
                 settings: {option: "value"} },  
            // ...  
        };  
    };  
});
```


Editor API: Menu Interface

`add_menu(name, id , icon)`

<code>name</code>	Name for the menu
<code>id</code>	HTML ID for reference
<code>icon</code>	Bootstrap glyphicon

`remove_menu_button(id)`

<code>id</code>	HTML ID for menu/button
-----------------	-------------------------

`add_menu_button( /*args*/ )`

<code>name</code>	Name for the menu
<code>id</code>	HTML id for later reference
<code>icon</code>	Bootstrap glyphicon string ¹
<code>cb_string</code>	String of function call (no " permitted)
<code>parent_menu</code>	(optional) ID for parent menu

Editor API: Creating New Tabs

`new_tab(name, callback)`

<code>name</code>	Name for the new tab
<code>callback</code>	Function that is called with the new view's HTML ID (shown when tab is selected)

```
window.new_tab("My Tab",  
  function(editor_name) {  
    var newHTML = "<p>I'm in a tab!</p>";  
    $("#"+editor_name).html(newHTML);  
  }  
);
```

Editor API: Code Snippets

`add_snippet(snippet, trigger)`

`snippet` Cloud9 style snippet

`trigger` Text to trigger the auto-complete

```
window.add_snippet(  
    "(when ${1: (and ())} \n\  
    (${2}))" ,  
    "condeff"  
)
```

Editor API: File Chooser

```
register_file_chooser(name, settings)
setup_file_chooser(btnName, desc)
```

name	Slug or nickname for the chooser
settings	Object including <code>showChoice</code> and <code>selectChoice</code> functions
btnName	Name of the button for submission
desc	Description for the top of the dialog

```
window.register_file_chooser('planner',
{
  showChoice: function() {
    window.setup_file_chooser('Plan', 'Compute Plan');
    $('#plannerURL').val(window.solverURL);
  },
  selectChoice: findPlan // Called when selected
});
```

Editor API: Injecting CSS / Making Toast

```
inject_styles(css_style)
```

`css_style` String of CSS to be included

```
window.inject_styles(  
    ".some-divs { float: left; }\n    #some-other-div { padding: 13px; }"  
)
```

```
window.toastr.success("Hurray!")  
window.toastr.info("Things are happening...")  
window.toastr.warning("Uh oh.")  
window.toastr.error("I give up :(")
```

5min Break...

...plugin from
start-to-finish