1 Narrative Documentation
   1.1 Quick start .................................................. 3
   1.2 The ‘pyramid’ command ................................... 3
   1.3 Settings ......................................................... 4
   1.4 Authentication ............................................... 6
   1.5 Profiling ......................................................... 6

2 Api Documentation
   2.1 pyramid_cubicweb ........................................... 9

3 Change History
   3.1 Pyramid Cubicweb Changes ................................. 15

4 Indices and tables ................................................. 17

Python Module Index ............................................... 19
Pyramid CubicWeb is an attempt to rebase the CubicWeb framework on pyramid.

It can be used in two different ways:

- Within CubicWeb, through the ‘pyramid’ cube and the *pyramid command*. In this mode, the Pyramid CubicWeb replaces some parts of CubicWeb and make the pyramid api available to the cubes.
- Within a pyramid application, it provides easy access to a CubicWeb instance and registry.
1.1 Quick start

1.1.1 From CubicWeb

- Install everything (here with pip, possibly in a virtualenv):
  ```
  pip install pyramid-cubicweb cubicweb-pyramid pyramid_debugtoolbar
  ```

- Make sure CubicWeb is in user mode:
  ```
  export CW_MODE=--user
  ```

- Create a CubicWeb instance, and install the ‘pyramid’ cube on it (see Set-up of a CubicWeb environment for more details on this step):
  ```
  cubicweb-ctl create pyramid myinstance
  ```

  **Warning:** You must allow anonymous access.

- Edit your `~/etc/cubicweb.d/myinstance/all-in-one.conf` and set values for `pyramid-auth-secret` and `pyramid-session-secret`.

- Start the instance with the ‘pyramid’ command instead of ‘start’:
  ```
  cubicweb-ctl pyramid --debug myinstance
  ```

1.1.2 In a pyramid application

Coming soon.

1.2 The ‘pyramid’ command

The ‘pyramid’ command is a replacement for the ‘start’ command of `cubicweb-ctl` tool. It provides the same options and a few other ones.

**Note:** The ‘pyramid’ command is provided by the `pyramid` cube.
1.2.1 Options

--no-daemon
  Run the server in the foreground.

--debug-mode
  Activate the repository debug mode (logs in the console and the debug toolbar). Implies --no-daemon

-D, --debug
  Equals to --debug-mode --no-daemon --reload

--reload
  Restart the server if any source file is changed

--reload-interval=RELOAD_INTERVAL
  Interval, in seconds, between file modifications checks [current: 1]

-l <log level>, --loglevel=<log level>
  Set the loglevel. debug if -D is set, error otherwise

-p, --profile
  Enable profiling. See Profiling.

--profile-output=PROFILE_OUTPUT
  Profiling output file (default: “program.prof”)

--profile-dump-every=N
  Dump profile stats to ouput every N requests (default: 100)

1.3 Settings

1.3.1 Cubicweb Settings

Pyramid CubicWeb will make use of the following configuration entries if found in the cubicweb configuration (a.k.a. all-in-one.conf):

> Warning: These settings requires the pyramid cube to be enabled on the instance.

pyramid-session-secret
  Secret phrase to sign the session cookie
  Used by pyramid_cubicweb.session.includeme() to configure the default session factory.

pyramid-session-secret = <some very secret passphrase>

pyramid-auth-secret
  Secret phrase to sign the authentication cookie
  Used by pyramid_cubicweb.auth.includeme() to configure the default authentication policy.

pyramid-auth-secret = <some other very secret passphrase>

1.3.2 Pyramid Settings

If a pyramid.ini file is found in the instance home directory (where the all-in-one.conf file is), its [main] section will be read and used as the settings of the pyramid Configurator.

This configuration file is almost the same as the one read by pserve, which allow to easily add any pyramid extension and configure it.

A typical pyramid.ini file is:
The Pyramid CubicWeb specific configuration entries are:

- **cubicweb.includes (list)**
  
  Same as `pyramid.includes`, but the includes are done after the cubicweb specific registry entries are initialized.
  
  Useful to include extensions that requires these entries.

- **cubicweb.bwcompat (bool)**
  
  (True) Enable/disable backward compatibility. See `pyramid_cubicweb.bwcompat`.

- **cubicweb.defaults (bool)**
  
  (True) Enable/disable defaults. See `pyramid_cubicweb.defaults`.

- **cubicweb.profile (bool)**
  
  (False) Enable/disable profiling. See Profiling.

- **cubicweb.auth.update_login_time (bool)**
  
  (True) Add a `pyramid_cubicweb.auth.UpdateLoginTimeAuthenticationPolicy` policy, that update the CWUser.login_time attribute when a user login.

- **cubicweb.auth.authtkt (bool)**
  
  (True) Enables the 2 cookie-base auth policies, which activate/deactivate depending on the `persistent` argument passed to `remember`.

  The first policy handles session authentication. It doesn’t get activated if `remember()` is called with `persistent=False`:

  - **cubicweb.auth.authtkt.session.cookie_name (str)**
    
    ('auth_tkt') The cookie name. Must be different from the persistent authentication cookie name.

  - **cubicweb.auth.authtkt.session.timeout (int)**
    
    1200. Cookie timeout.

  - **cubicweb.auth.authtkt.session.reissue_time (int)**
    
    120. Reissue time.

  The second policy handles persistent authentication. It doesn’t get activated if `remember()` is called with `persistent=True`:

  - **cubicweb.auth.authtkt.persistent.cookie_name (str)**
    
    ('auth_tkt') The cookie name. Must be different from the session authentication cookie name.

  - **cubicweb.auth.authtkt.persistent.max_age (int)**
    
    (30 days) Max age in seconds.
**cubicweb.auth.authtkt.persistent.reissue_time** *(int)*
(1 day) Reissue time in seconds.

**cubicweb.auth.groups_principals** *(bool)*
(True) Setup a callback on the authentication stack that inject the user groups in the principals.

## 1.4 Authentication

### 1.4.1 Overview

A default authentication stack is provided by the `pyramid_cubicweb.auth` module, which is included by `pyramid_cubicweb.default`.

The authentication stack is built around `pyramid_multiauth`, and provides a few default policies that reproduce the default cubicweb behavior.

*Note:* Note that this module only provides an authentication policy, not the views that handle the login form. See `pyramid_cubicweb.login`.

### 1.4.2 Customize

The default policies can be individually deactivated, as well as the default authentication callback that returns the current user groups as principals.

The following settings can be set to `False`:

- `cubicweb.auth.update_login_time`. Activate the policy that update the user `login_time` when `remember` is called.
- `cubicweb.auth.authtkt` and all its subvalues.
- `cubicweb.auth.groups_principals`

Additional policies can be added by accessing the MultiAuthenticationPolicy instance in the registry:

```python
mypolicy = SomePolicy()
authpolicy = config.registry["cubicweb.authpolicy"]
authpolicy._policies.append(mypolicy)
```

## 1.5 Profiling

Profiling of requests by the pyramid debug toolbar can be a little restrictive when a specific url needs thin profiling that includes the whole pyramid dispatch.

Pyramid CubicWeb provides facilities to profile requests as a wsgi middleware, and a few views that facilitate profiling of basic features.

The views and the wsgi middleware are activated when the ‘profile’ option is given. This can be done on the command line (`cubicweb-ctl pyramid --profile`) or in the *Pyramid Settings*.

### 1.5.1 Views

The following routes and corresponding views are provided when profiling is on:

- `/_profile/ping`: Reply ‘ping’ without doing anything else. See also `pyramid_cubicweb.profile.ping()`.
1.5.2 Typical Usage

Let's say we want to measure the cost of having a.cnx.

- Start the application with profile enabled:
  
  ```
  $ cubicweb-ctl pyramid --no-daemon --profile --profile-dump-every 100
  ```

- Use `ab` or any other http benchmark tool to throw a lot of requests:
  
  ```
  $ ab -c 1 -n 100 http://localhost:8080/_profile/cnx
  ```

- Analyse the results. I personally fancy **SnakeViz**:
  
  ```
  $ snakeviz program.prof
  ```
Api Documentation

2.1 pyramid_cubicweb

pyramid_cubicweb.make_cubicweb_application(cwconfig, settings=None)

Create a pyramid-based CubicWeb instance from a cubicweb configuration.

Parameters
- cwconfig – A CubicWeb configuration

Returns
A Pyramid config object

pyramid_cubicweb.wsgi_application_from_cwconfig(cwconfig, profile=False, profile_output=None, profile_dump_every=None)

Build a WSGI application from a cubicweb configuration

Parameters
- cwconfig – A CubicWeb configuration
- profile – Enable profiling. See Profiling.
- profile_output – Profiling output filename. See Profiling.
- profile_dump_every – Profiling number of requests before dumping the stats. See Profiling.

Returns
A fully operationnal WSGI application

pyramid_cubicweb.wsgi_application(instance_name=None, debug=None)

Build a WSGI application from a cubicweb instance name

Parameters
- instance_name – Name of the cubicweb instance (optional). If not provided, CW_INSTANCE must exists.
- debug – Enable/disable the debug mode. If defined to True or False, overrides CW_DEBUG.

The following environment variables are used if they exist:

CW_INSTANCE
A CubicWeb instance name.

CW_DEBUG
If defined, the debugmode is enabled.

The function can be used as an entry-point for third-party wsgi containers. Below is a sample uWSGI configuration file:
2.1.1 pyramid_cubicweb.auth

pyramid_cubicweb.auth.includeme(config)
Activate the CubicWeb AuthTkt authentication policy.

Usually called via config.include('pyramid_cubicweb.auth').

See also pyramid_cubicweb.defaults

class pyramid_cubicweb.auth.UpdateLoginTimeAuthenticationPolicy
Bases: object
An authentication policy that update the user last_login_time.

The update is done in the ‘remember’ method, which is called by the login views login.

Usually used via includeme().

2.1.2 pyramid_cubicweb.authplugin

2.1.3 pyramid_cubicweb.bwcompat

pyramid_cubicweb.bwcompat.includeme(config)
Set up a tween app that will handle the request if the main application raises a HTTPNotFound exception.

This is to keep legacy compatibility for cubes that makes use of the cubicweb urlresolvers.

It provides, for now, support for cubicweb controllers, but this feature will be reimplemented separately in a less compatible way.

It is automatically included by the configuration system, but can be disabled in the Pyramid Settings:

cubicweb.bwcompat = no

class pyramid_cubicweb.bwcompat.PyramidSessionHandler(appli)
A CW Session handler that rely on the pyramid API to fetch the needed informations.

It implements the cubicweb.web.application.CookieSessionHandler API.

class pyramid_cubicweb.bwcompat.CubicWebPyramidHandler(appli)
A Pyramid request handler that rely on a cubicweb instance to do the whole job

Parameters appli – A CubicWeb ‘Application’ object.

__call__(request)
Handler that mimics what CubicWebPublisher.main_handle_request and CubicWebPublisher.core_handle do

class pyramid_cubicweb.bwcompat.TweenHandler(handler, registry)
A Pyramid tween handler that submit unhandled requests to a Cubicweb handler.

The CubicWeb handler to use is expected to be in the pyramid registry, at key ‘cubicweb.handler’.
2.1.4 pyramid_cubicweb.core

`pyramid_cubicweb.core.includeme(config)`
Enables the core features of Pyramid CubicWeb.

Automatically called by the ‘pyramid’ command, or via `config.include('pyramid_cubicweb.code')`. In the later case, the following registry entries must be defined first:

- ‘cubicweb.config’ A cubicweb ‘config’ instance.
- ‘cubicweb.repository’ The corresponding cubicweb repository.
- ‘cubicweb.registry’ The vreg.

`pyramid_cubicweb.core.cw_to_pyramid(*args, **kwds)`
Context manager to wrap a call to the cubicweb API.

All CW exceptions will be transformed into their pyramid equivalent. When needed, some CW response bits may be converted too (mainly headers)

`pyramid_cubicweb.core.render_view(request, vid, **kwargs)`
Helper function to render a CubicWeb view.

**Parameters**

- `request` – A pyramid request
- `vid` – A CubicWeb view id
- `**kwargs` – Keyword arguments to select and instantiate the view

**Returns**

The rendered view content

`pyramid_cubicweb.core.repo_connect(request, repo, eid)`
A lightweight version of `cubicweb.server.repository.Repository.connect()` that does not keep track of opened sessions, removing the need of closing them

`pyramid_cubicweb.core.get_principals(login, request)`
Returns the group names of the authenticated user.

This function is meant to be used as an authentication policy callback.

It also pre-open the cubicweb session and put it in `request._cw_cached_session` for later usage by `_cw_session()`.

**Note:** If the default authentication policy is not used, make sure this function gets called by the active authentication policy.

**Parameters**

- `login` – A cubicweb user eid
- `request` – A pyramid request

**Returns** A list of group names

`class pyramid_cubicweb.core.CubicWebPyramidRequest(request)`
Bases: `cubicweb.web.request.ConnectionCubicWebRequestBase`

A CubicWeb request that only wraps a pyramid request.

**Parameters**

- `request` – A pyramid request

**message**

Returns a ‘<br>’ joined list of the cubicweb current message and the default pyramid flash queue messages.

`pyramid_cubicweb.core._cw_session(request)`
Obtains a cw session from a pyramid request
Parameters `request` – A pyramid request

Returns type `cubicweb.server.session.Session`

Not meant for direct use, use `request.cw_session` instead.

`pyramid_cubicweb.core._cw_cnx(request)`
Obtains a cw session from a pyramid request

The connection will be committed or rolled-back in a request finish callback (this is temporary, we should make use of the transaction manager in a later version).

Not meant for direct use, use `request.cw_cnx` instead.

Parameters `request` – A pyramid request

Returns type `cubicweb.server.session.Connection`

`pyramid_cubicweb.core._cw_request(request)`
Obtains a CubicWeb request wrapper for the pyramid request.

Parameters `request` – A pyramid request

Returns type `CubicWebPyramidRequest`

Not meant for direct use, use `request.cw_request` instead.

### 2.1.5 `pyramid_cubicweb.defaults`

Defaults for a classical CubicWeb instance.

`pyramid_cubicweb.defaults.include_me(config)`
Enable the defaults that make the application behave like a classical CubicWeb instance.

The following modules get included:

- `pyramid_cubicweb.session`
- `pyramid_cubicweb.auth`
- `pyramid_cubicweb.login`

It is automatically included by the configuration system, unless the following entry is added to the Pyramid Settings:

```
cubicweb.defaults = no
```

### 2.1.6 `pyramid_cubicweb.login`

Provide login views that reproduce a classical CubicWeb behavior

`pyramid_cubicweb.login.include_me(config)`
Create the ‘login’ route (’/login’) and load this module views

**Views**

`pyramid_cubicweb.login.login_form(request)`
Default view for the ‘login’ route.

Display the ‘login’ CubicWeb view, which is should be a login form
pyramid_cubicweb.login.login_password_login(request)
Handle GET/POST of __login__/__password on the ‘login’ route.
The authentication itself is delegated to the CubicWeb repository.

Request parameters:

Parameters

• __login – The user login (or email if allow-email-login is on).
• __password – The user password
• __setauthcookie – (optional) If defined and equal to ‘1’, set the authentication cookie maxage to 1 week.

If not, the authentication cookie is a session cookie.

pyramid_cubicweb.login.login_already_loggedin(request)
‘login’ route view for Authenticated users.

Simply redirect the user to ‘/’.

2.1.7 pyramid_cubicweb.profile

Tools for profiling.

See Profiling.

Views

pyramid_cubicweb.profile.ping(request)
View that handle ‘/_profile/ping’
It simply reply ‘ping’, without requiring connection to the repository. It is a useful as a comparison point to evaluate the actual overhead of more costly views.

pyramid_cubicweb.profile.cnx(request)
View that handle ‘/_profile/cnx’
Same as ping(), but it first ask for a connection to the repository. Useful to evaluate the overhead of opening a connection.

WSGI

pyramid_cubicweb.profile.wsgi_profile(app, filename='program.prof', dump_every=50)
A WSGI middleware for profiling
It enable the profiler before passing the request to the underlying application, and disable it just after.
The stats will be dumped after dump_every requests

Parameters

• filename – The filename to dump the stats to.
• dump_every – Number of requests after which to dump the stats.

2.1.8 pyramid_cubicweb.session

pyramid_cubicweb.session.include_me(config)
Activate the CubicWeb session factory.

Usually called via config.include(‘pyramid_cubicweb.auth’).
See also `pyramid_cubicweb.defaults`

**`pyramid_cubicweb.session.CWSessionFactory`**

```python
pyramid_cubicweb.session.CWSessionFactory(
    secret,  # cookie_name='session',
    max_age=None,  # path='/',
    domain=None,  # secure=False, httponly=False,
    set_on_exception=True,  # timeout=1200,
    reissue_time=120,  # hashalg='sha512',
    salt='pyramid.session.',
    serializer=None,
)
```

A pyramid session factory that store session data in the CubicWeb database.

Storage is done with the ‘CWSession’ entity, which is provided by the ‘pyramid’ cube.

**Warning:** Although it provides a sane default behavior, this session storage has a serious overhead because it uses RQL to access the database. Using pure SQL would improve a bit (it is roughly twice faster), but it is still pretty slow and thus not an immediate priority.

It is recommended to use faster session factory (`pyramid_redis_sessions` for example) if you need speed.

### 2.1.9 `pyramid_cubicweb.tools`

Various tools.

**Warning:** This module should be considered as internal implementation details. Use with caution, as the API may change without notice.

**`pyramid_cubicweb.tools.includeme(config)`**

Start the cache maintenance loop task.

Automatically included by `pyramid_cubicweb.make_cubicweb_application()`.

**`pyramid_cubicweb.tools.clone_user(repo, user)`**

Clone a CWUser instance.

**Warning:** The returned clone is detached from any cnx. Before using it in any way, it should be attached to a cnx that has not this user already loaded.

**`pyramid_cubicweb.tools.cnx_attach_entity(cnx, entity)`**

Attach an entity to a cnx.

**`pyramid_cubicweb.tools.cached_build_user(repo, eid)`**

Cached version of `cubicweb.server.repository.Repository._build_user()`

**`pyramid_cubicweb.tools.clear_cache()`**

Clear the user cache
3.1 Pyramid Cubicweb Changes

3.1.1 0.3.1 (2015-06-18)

• debian: add python-wsgicors dependency (issue 4751889).
• Handle absence of anonymous user (issue 4751862).

3.1.2 0.3.0 (2015-05-11)

• Reorganize the authentication stack around pyramid_multiauth (issue 4985962).
• Implement the CW message API on top of the Pyramid Flash Messages (issue 5298654).
• Don’t commit ‘uncommitable’ connexions anymore (issue 5343870).
• Debug mode enables pyramid.reload_templates.
• Testcases can override pyramid settings (issue 5307426).
• pyramid_debugtoolbar is not mandatory anymore (issue 5310434).
• Add unit tests (coverage 79%).
• Performance improvements (issue 4891437 & issue 4870347).
• Documentation improvements
• Set response headers on exceptions (issue 4939219).
• Rename the package name from ‘pyramid_cubicweb’ to ‘pyramid-cubicweb’, for consistency with other pyramid extensions.

3.1.3 0.2.1 (2015-01-23)

• Fix cors ‘methods’ and ‘headers’ parameters passing (issue 4849874).

3.1.4 0.2.0 (2015-01-21)

• Create a documentation (issue 4849313)
• Fix cors ‘origin’ parameter passing (issue 4783343)
• Fix configuration loading when ‘cubicweb.includes’ is not set (issue 4849314)
• Move auth-related code to pyramid_cubicweb/auth.
• Add profiling tools
• Cleanups

3.1.5 0.1.3 (2014-12-08)
• Fix cookies max_age (issue 4731764)

3.1.6 0.1.2 (2014-11-15)
• Fix excessive rollbacks on HTTPSuccessful or HTTPRedirection (issue 4566482)

3.1.7 0.1.1 (2014-11-02)
• Have CWUser.last_login_time properly updated (issue 4549891)

3.1.8 0.1.0 (2014-10-23)
Initial release
• Provides a pyramid-based authentication and session management for a cubicweb instance.
• Run a cubicweb instance as a pyramid application
Indices and tables

- genindex
- modindex
- search
p

pyramid_cubicweb, 9
pyramid_cubicweb.auth, 10
pyramid_cubicweb.bwcompat, 10
pyramid_cubicweb.core, 11
pyramid_cubicweb.defaults, 12
pyramid_cubicweb.login, 12
pyramid_cubicweb.profile, 13
pyramid_cubicweb.session, 13
pyramid_cubicweb.tools, 14
Symbols

--debug-mode
cubicweb-ctl-pyramid command line option, 4
--no-daemon
cubicweb-ctl-pyramid command line option, 4
--profile-dump-every=N
cubicweb-ctl-pyramid command line option, 4
--profile-output=PROFILE_OUTPUT
cubicweb-ctl-pyramid command line option, 4
-reload
cubicweb-ctl-pyramid command line option, 4
-reload-interval=RELOAD_INTERVAL
cubicweb-ctl-pyramid command line option, 4
-D, --debug
cubicweb-ctl-pyramid command line option, 4
-l <log level>, --loglevel=<log level>
cubicweb-ctl-pyramid command line option, 4
-p, --profile
cubicweb-ctl-pyramid command line option, 4
__call__() (pyramid_cubicweb.bwcompat.CubicWebPyramidHandler
  method), 10
_cw_cnx() (in module pyramid_cubicweb.core), 12
_cw_request() (in module pyramid_cubicweb.core), 12
_cw_session() (in module pyramid_cubicweb.core), 11
C
cached_build_user() (in module pyramid_cubicweb.tools), 14
clear_cache() (in module pyramid_cubicweb.tools), 14
copy_user() (in module pyramid_cubicweb.tools), 14
cnx() (in module pyramid_cubicweb_profile), 13
cnx_attach_entity() (in module pyramid_cubicweb.tools), 14
configuration value
cubicweb.auth.authtkt(bool), 5
cubicweb.auth.authtkt.persistent.cookie_name(str), 5
cubicweb.auth.authtkt.persistent.max_age(int), 5
cubicweb.auth.authtkt.persistent.reissue_time(int), 5
cubicweb.auth.authtkt.session.cookie_name(str), 5
cubicweb.auth.authtkt.session.reissue_time(int), 5
cubicweb.auth.authtkt.session.timeout(int), 5
cubicweb.auth.groups_principals(bool), 6
cubicweb.auth.update_login_time(bool), 5
cubicweb.bwcompat(bool), 5
cubicweb.defaults(bool), 5
cubicweb.includes(list), 5
cubicweb.profile(bool), 5
pyramid-auth-secret, 4
pyramid-session-secret, 4
cubicweb-ctl-pyramid command line option
--debug-mode, 4
--no-daemon, 4
--profile-dump-every=N, 4
--profile-output=PROFILE_OUTPUT, 4
-reload, 4
-reload-interval=RELOAD_INTERVAL, 4
-D, --debug, 4
-l <log level>, --loglevel=<log level>, 4
-p, --profile, 4
cubicweb.auth.authtkt(bool)
configuration value, 5
cubicweb.auth.authtkt.persistent.cookie_name(str)
configuration value, 5
cubicweb.auth.authtkt.persistent.max_age(int)
configuration value, 5
cubicweb.auth.authtkt.persistent.reissue_time(int)
configuration value, 5
cubicweb.auth.authtkt.session.cookie_name(str)
configuration value, 5
cubicweb.auth.authtkt.session.reissue_time(int)
configuration value, 5
cubicweb.auth.authtkt.session.timeout(int)
configuration value, 5
cubicweb.auth.groups_principals(bool)
configuration value, 6
cubicweb.auth.update_login_time(bool)
configuration value, 5
cubicweb.bwcompat(bool)
configuration value, 5
cubicweb.defaults(bool)
configuration value, 5
cubicweb.includes(list)
configuration value, 5
cubicweb.profile(bool)
configuration value, 5
CubicWebPyramidHandler (class in pyramid_cubicweb.bwcompat), 10
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
</tr>
<tr>
<td>CW_DEBUG</td>
</tr>
<tr>
<td>CW_INSTANCE</td>
</tr>
<tr>
<td>G</td>
</tr>
<tr>
<td>I</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>L</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>M</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>U</td>
</tr>
<tr>
<td>W</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>