PLATINUMSOFTWARE

PlatSoft Developer Manual

Table of Contents

1. GRAPE

1.1 Structure of a Grape Application

TODO

1.2 Authentication and access control

Grape manages authentication, sessions, users, passwords and provide access control. Sessions are tracked using a session ID. Session IDs can be obtained by making a call to ***POST /grape/login***, providing a valid username and password. On success, this call will return with the session ID, the user's ID, username, roles and employee GUID. In subsequent calls, the session ID is sent to the server using a custom header ***X-SessionID*** in the HTTP requests. Before an API call is executed, Grape will check the validity of the session ID, and do access control on the path against the user's roles.

Users and user-related information is stored in `grape.user`. Users can belong to one or more access roles, stored in `grape.access_path`. The link-table for these are `grape.user_role`.

Grape includes the following roles and access paths by default:

ROLE	DESCRIPTION	PATHS ALLOWED
guest	No or invalid login	• /grape/login
all	All logged in users	• /lookup/* • /grape/list • /grape/api_list
admin	Administrator	• * (all paths allowed)

1.3 API Calls

A typical Grape API call consists of:

1. A call to register the route in the Node layer
2. A call to connect this route to a function in PostgreSQL
3. A PostgreSQL function, taking in one JSON value and returning a JSON object
4. A PostgreSQL function containing the business logic for the call
5. If applicable, this call should be added to Grape's list of allowed routes

For example, let's create an API call to calculate the square root of a value in the database:

1. When Grape starts up, it reads a directory containing API files. This directory is controlled by setting the ***api_directory*** option.
2. In this directory, add a file to register this API call in.

```
exports = module.exports = function(app) {
    // register the route
    app.get("/maths/sqrt/:input", api_maths_sqrt);
}
function api_maths_sqrt (req, res)
{
    // call the stored procedure for this API call
```

```
    res.locals.db.json_call("maths_sqrt", // the name of the PL/pgSQL function
        {input: req.params.input}, // JSON object as input for this function
        null, // Optional callback (not used here)
        {response: res} // Send the response to res
    );
}
```

1.4 Sending emails from Grape

The following needs to be set up in order to send emails from within SQL functions:

1. smtp settings in config
2. email_template_directory in config containing templates

1.4.1 send_email

Call ***grape.send_email (to, template, data)*** to send an email. The following parameters are needed:

- ***to*** Email address of receiver
- ***template*** Template name
- ***data*** Template data

1.4.2 Email templates

Email templates live in the email_template_directory defined in the app's config. The template engine reads 4 files (***templatename*** is replaced by the name provided when calling grape.send_email):

- ***templatename.subject*** - To generate the subject
- ***templatename.text*** - To generate the plain-text body of the email
- ***templatename.html*** - To generate the HTML body of the email
- ***templatename.attachments*** - To generate a list of attachments to include in the email

The data sent to ***grape.send_email (to, template, data)*** is accessible inside the template files. For example, if an email is called with the following data:

```
{ "firstname": "Piet" }
```

The field firstname is accessable inside of the templates using `<%= firstname %>`

1.5 Known Grape settings

NAME	DESCRIPTION	DEFAULT VALUE
hash_passwords	Indicate whether passwords in grape.user is hashed	false
allow_default_paths	If a path is not found and this setting is true, access will be granted	false
grape_version	Current Grape version	

NAME	DESCRIPTION	DEFAULT VALUE
product_name	Name of the current system	
product_uuid	Unique identifier for the product/system	
product_version	Product version	
data_upload_schema	Default schema for data import tables	grape
disable_passwords	If true, authentication will not check whether the password is correct	false

1.6 Grape config file

The following options are recognized in the config passed to Grape:

NAME	DESCRIPTION	DEFAULT VALUE
dburi	DB connection settings	
guest_dburi	DB connection settings for guest users	
api_directory		
db_definition	Array containing directories with DB definitions. Subdirectories schema, function, view and data will be traversed when recreating	
sql_dirs	Array containing directories with DB definitions. All subdirectories will be recursively read	
pg_temp_directory	Path to a directory to which PostgreSQL has write access	
port	Port on which the UI will be available	
http_port	If this is set, and HTTPS is enabled (use_https), then a normal HTTP server wil listen on this port	
public_directory	Directory containing public files	
public_directories	List of directories containing public files	
debug		true
maxsockets	Controls the maximum number of sockets supported	500

NAME	DESCRIPTION	DEFAULT VALUE
bordeaux_config_file	Path to Bordeaux config file	__dirname + '/bordeaux_config.json'
document_store	Path to document store	
use_https	Enable or disable HTTPS. sslkey and sslcert need to be set up correctly	false
session_management	Enable or disable session management	true
use_https		
smtp	SMTP settings for GrapeMailer	
server_timeout	The number of milliseconds of inactivity before a socket is presumed to have timed out	50000
sslkey	Path to private SSL key file	__dirname + '/cert/private.pem'
sslcert	Path to private SSL public certificate	__dirname + '/cert/public_nopass.pem'
hr_system	URL to get access to the Bordeaux system running on Savanna HR system	https://192.168.50.86:3999/
email_template_directory	Path to email templates (See GrapeMailer for more information)	__dirname + '/email_templates'

1.7 Standardized Error Codes

CODE	DESCRIPTION
-1	Unknown Error
-2	Permission Denied
-3	Invalid Input
-5	Requested data not found
-99	Database Error