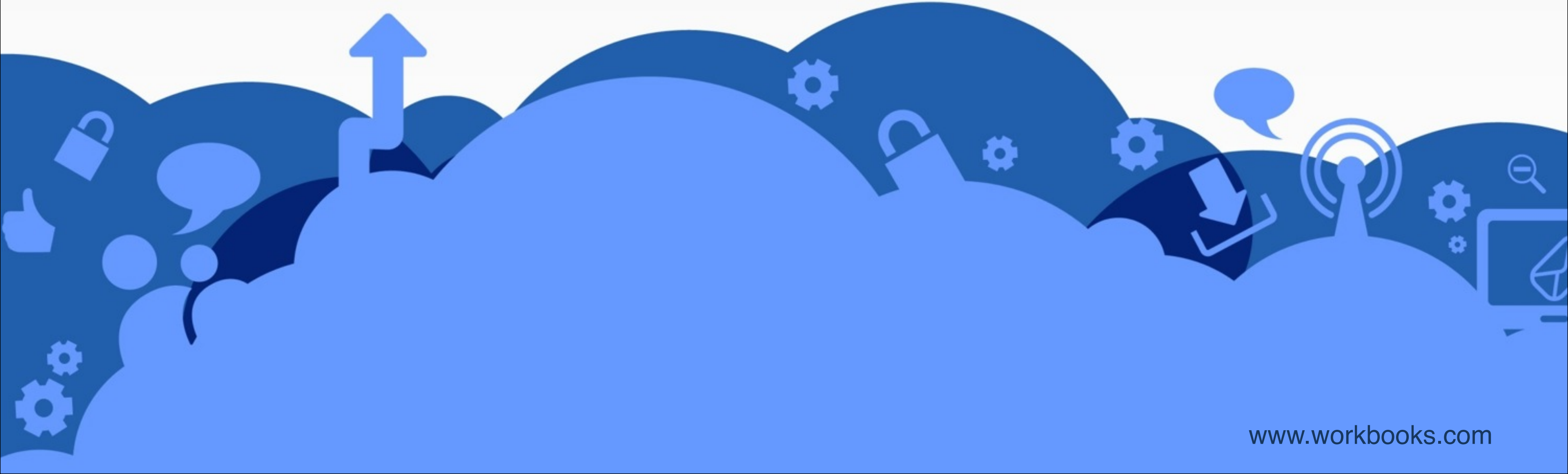
Workbooks Developer Training

March 2014



www.workbooks.com

Agenda

- Introduction to Workbooks and its programming model
- What you can do with the API
- How to use the API
  - External access or the Process Engine?
  - PHP in 30 minutes
  - Security Model
  - Get, Create, Update and Delete with the API. Metadata
- The Process Engine, process types, how processes are run
- Special APIs, Reporting, Emailing
- Writing supportable scripts
- Getting support from Workbooks

Introduction: Why use the API?

Some examples:

- Email-to-Case: monitor a mailbox, create and update support cases
- MailChimp, Constant Contact, dotMailer, HubSpot …
- Sagelink, OneSaaS
- Outlook Connector
- Mobile Client
- Creating many order line items to reflect a delivery schedule
- Calculate field values
- Sales Lead categorisation, analysis and allocation

API not required:

- Simple lead or case capture (use web-to-case)
- Generating a PDF (use PDF templates) or a templated email
- Simple workflow using custom page layouts and assignment
- Data Import
- Reporting

Introduction: What is the API?

- API – ‘Application Programmatic Interface’
  - i.e. an interface enabling software to interact with Workbooks.
- A set of web services delivered over SSL (https)
- Stateless, client/server
- RESTful – create, read, update, delete
- Batched
- JSON, UTF-8

How to call the API: Wire Protocol or Binding?

- Wire Protocol = JSON-encoded HTTP requests
  - Can be complex
  - Documented at
    - http://www.workbooks.com/api-developer-guide
  - No restriction on which language is used.
- Bindings hide much of the complexity
  - PHP binding and example code is on github (please feel free to contribute) at
    - https://github.com/workbooks/client_lib/tree/master/php
    - PHP used by the process engine
      - PHP is widely-understood and open-source.
      - Lots of systems have documented PHP APIs.
  - Others to come: Java, .NET(C#) …

Wire Protocol versus Binding

e.g. Fetching a couple of fields from a record by ID.

```
/crm/people.api?_ff%5B%5D=id&_ft%5B%5D=eq&_fc%5B
%5D=21458&&_start=0&_limit=1&_select_columns%5B%5D=id&_select_columns%5B
%5D=main_location%5Bcountry%5D&_select_columns%5B%5D=main_location%5Bemail%5D
```

```
$response = $workbooks->assertGet('crm/people',
  array(
    '_filters[]' => array('id', 'eq', 21458),
    '_limit' => 1,
    '_start' => 0,
    '_select_columns[]' => array(
         'id', 'main_location[country]', 'main_location[email]'),
  )
);
```

Modifying records with the wire protocol is more complex: URL encoding, _authenticity_token etc.

In both cases, need to check the response. assertGet() includes error checking.

Exercise: Workbooks Desktop network traffic

- Open your favourite web browser
- Reveal Developer Tools and show network traffic
- Login to the Workbooks Desktop
- Clear the the network traffic
- Open a landing page and an item
- Examine the .extjs and .wbjson traffic, especially the request and response headers

- This is **not** the Workbooks API. But it is close

- Note that it is https, compressed with gzip, JSON-based

Login to a test account:

username:
api.training@workbooks.com
password:
crmsuccess

How to call the API:
Where to run your code?

- Workbooks-hosted
  - The “Process Engine”.
  - Simpler, automates invocation, authentication and logging.
  - Not available if you are using a ‘Free’ licence.
- Externally
  - Host your code yourself.
  - Connect to Workbooks explicitly over HTTPS.
  - Authenticate using API Key or Username, password and database ID.
  - A little more flexible.
  - From an on-premises system, avoids most firewall issues.
  - The API is available to all Workbooks users.
- The Process Engine is used in this presentation for simplicity.

Workbooks Security Model: Authentication

- Username / password
  - not recommended for API clients unless a user is entering these in a UI
  - require database selection
  - require database selection
- API Keys
  - no password to expire
  - use a different API Key for each API client
  - easy to restrict by time or IP address
  - specific to a database
- Cookies
  - Authentication happens at /login.api which returns a Workbooks-Session cookie
  - Cookie value changes during session
  - Cookie should be sent with each request
  - Not needed if you pass an API Key with each request (slower and less efficient)
- Required: _application_name during login, user-agent header

Using the Wire Protocol with curl

- cURL is a command-line tool for requesting data with URLs
- Very flexible as a test/experimental tool
- Downloads for most platforms, from http://curl.haxx.se/
- Lots of examples in the Workbooks API Developers Guide, e.g.

```
curl -i -g -s --tlsv1 \
         -c '/tmp/cookiejar' \                       Store credentials for future calls
         -A 'XYZ plugin/1.2.3 (gzip)' \              User-agent string
         --compressed \                              Use gzip compression
         -X 'POST' \                                 HTTP POST (not GET)
         -d 'username=system_test@workbooks.com' \  A series of fields
         -d 'password=abc123' \
         -d 'client=api' \
         -d 'json=pretty' \
         https://secure.workbooks.com/login.api      The URL to send to
```

Exercise: Using curl

- Download curl.
- Cut-and-paste to run a couple of the examples from the Workbooks API Developer Guide
  - Login
  - Retrieve
  - Create

- Note that parameters to create, update, delete are arrays! Append [] to the field names you are changing.

Workbooks Security Model: Capabilities & Permissions

- Licenses, Modules, Capabilities
  - Most ‘controller-actions’ require specific capabilities
    - crm/people, index (or create, edit, delete)
  - Capabilities are assigned through Group membership
  - API clients often require more capabilities than the users who use them
- Permissions - per-record
  - Read / Modify / Delete / Change ownership and permission
  - Again, API clients often require more visibility and access to records than the users who use them
  - Set upon record creation or ownership-change according to configured rules

Databases

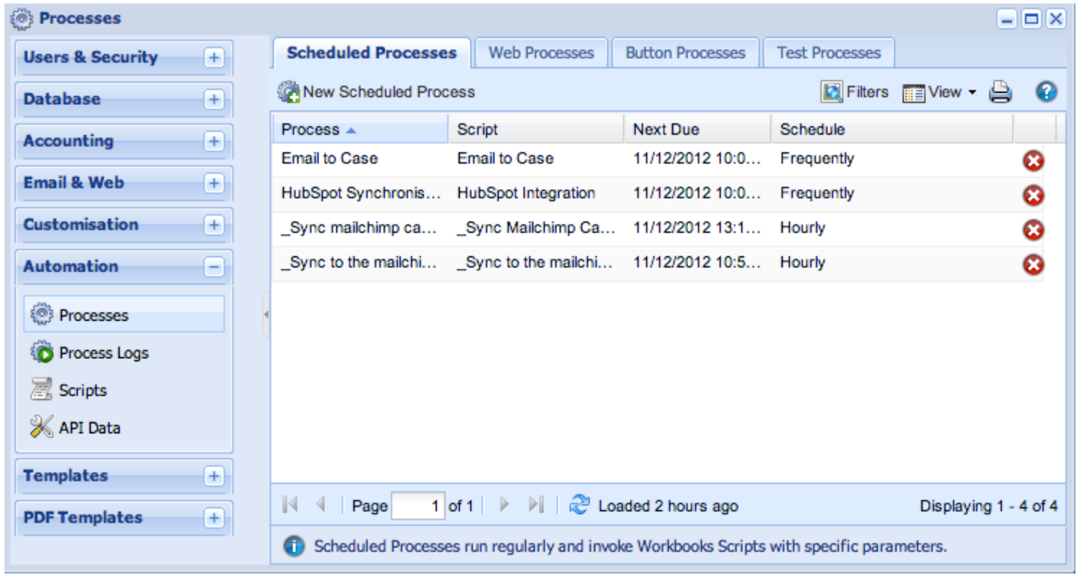
- Workbooks customers normally have more than one database
- Databases can be used for
  - backup,
  - staging / testing / sandboxing,
  - segregation of data (but permissions are more flexible)
- Choose database at login time
- Copy a database
  - creates a completely separate copy of all data
  - users are shared (as are passwords)
  - capabilities, group membership, api keys, web2lead keys etc are per-database
- Can export to SQL. ODBC access is not permitted.

Introducing the Process Engine

- Some glossary:
  - Script – a unit of code.
  - Processes invoke Scripts.
  - Process types:
    - Scheduled Process
    - Web Process
    - Process Button / on-Save Process
    - Test Process
  - Processes run on behalf of a user, with constraints.

Processes

- Processes invoke Scripts
- Scheduled
- Web
- Button (& onSave)
- Report
- Test



Process	Script	Next Due	Schedule
Email to Case	Email to Case	11/12/2012 10:0...	Frequently
HubSpot Synchronis...	HubSpot Integration	11/12/2012 10:0...	Frequently
_Sync mailchimp ca...	_Sync Mailchimp Ca...	11/12/2012 13:1...	Hourly
_Sync to the mailchi...	_Sync to the mailchi...	11/12/2012 10:5...	Hourly

Scheduled Processes run regularly and invoke Workbooks Scripts with specific parameters.

- Processes run as a user
  - User requires DB access.
  - Capabilities matter.

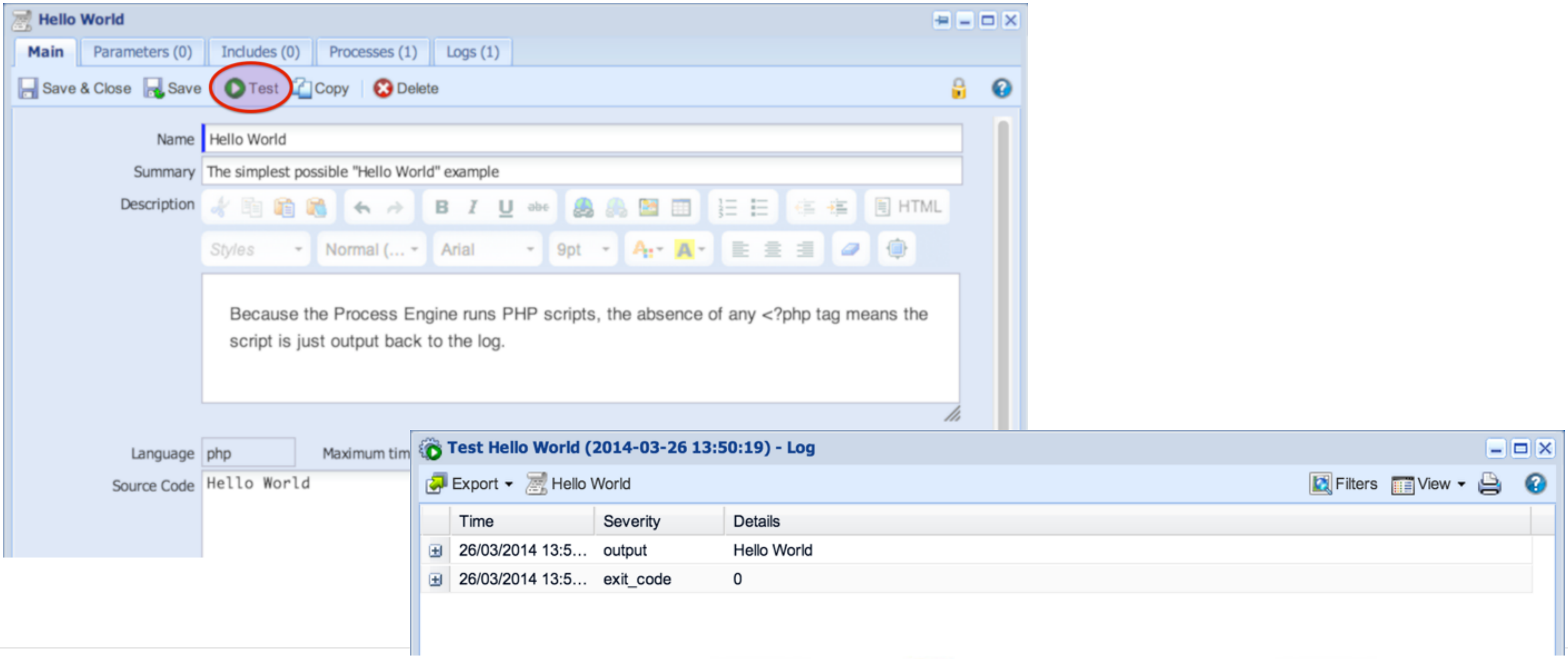
Workbooks.com

Test Process

- Useful for debugging simple scripts
- Created when first used
- Prompts for parameters

Example: Hello World!

- Is this cheating?



Name: Hello World

Summary: The simplest possible "Hello World" example

Description: Because the Process Engine runs PHP scripts, the absence of any <?php tag means the script is just output back to the log.

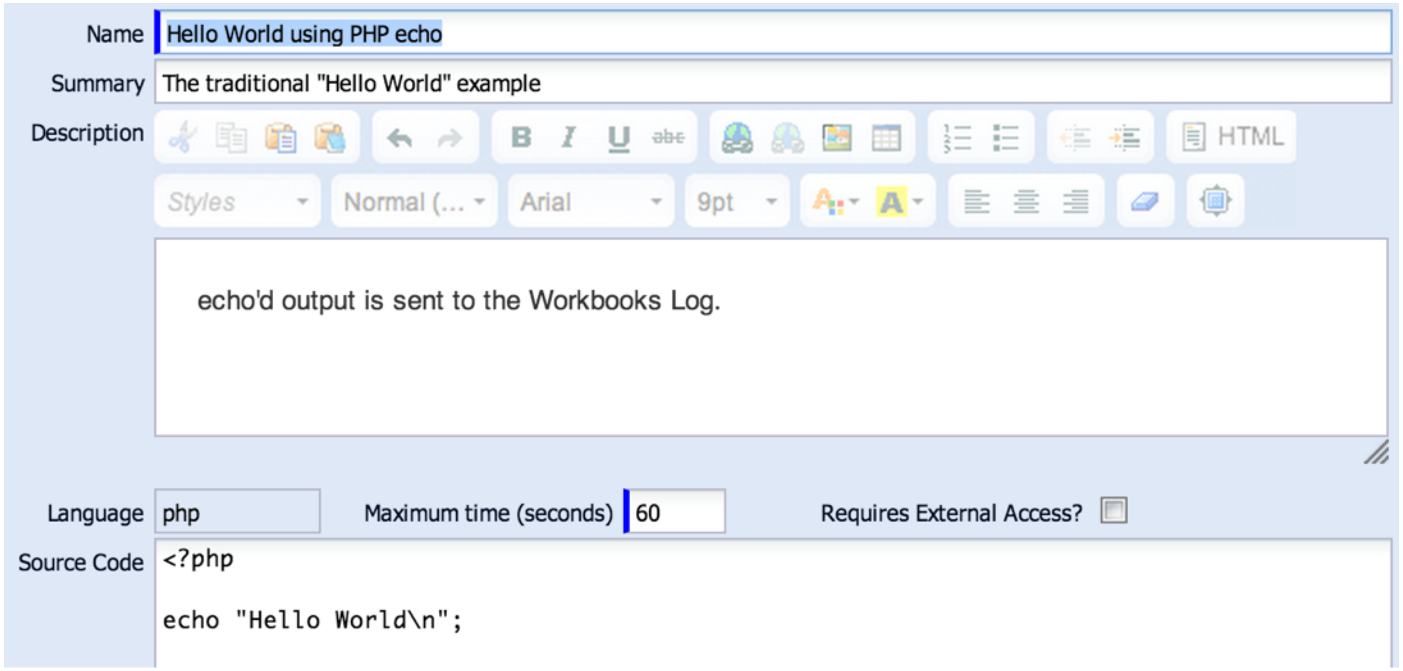
Language: php

Source Code:

```
Hello World
```

	Time	Severity	Details
⊞	26/03/2014 13:5...	output	Hello World
⊞	26/03/2014 13:5...	exit_code	0

Exercise: Hello World and phpinfo() in PHP



Name	Hello World using PHP echo
Summary	The traditional "Hello World" example
Description	echo'd output is sent to the Workbooks Log.
Language	php
Maximum time (seconds)	60
Requires External Access?	☐

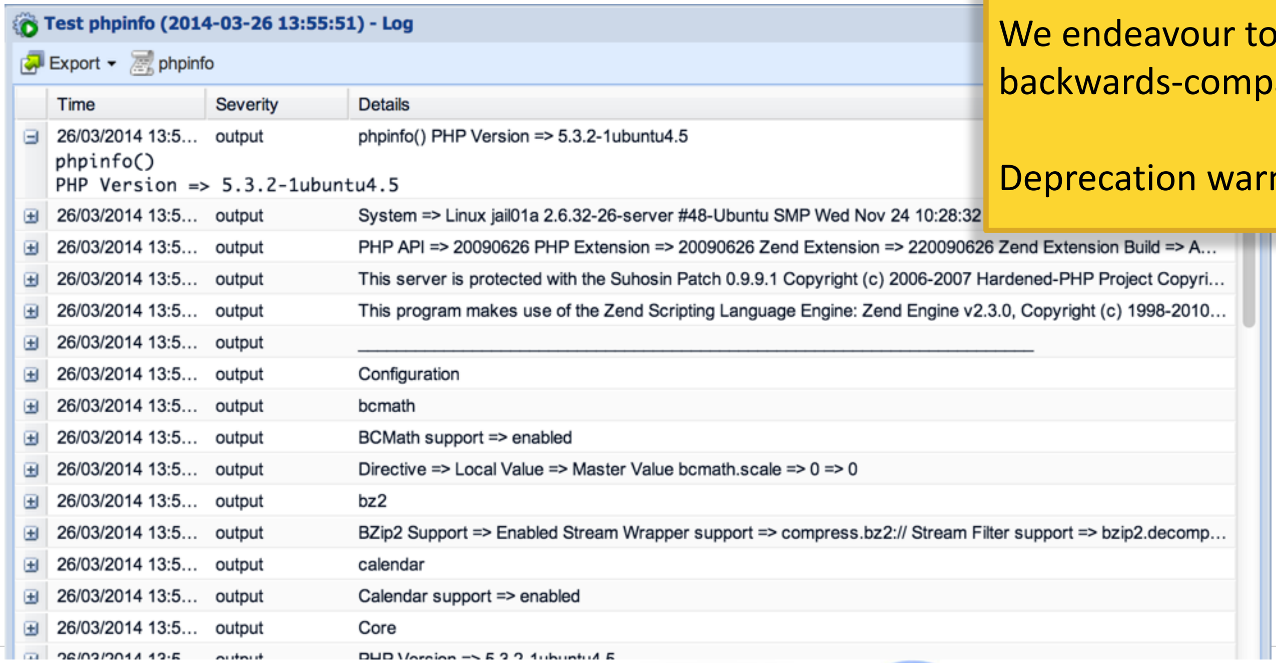
Source Code

```
<?php

echo "Hello World\n";
```

- Now try running the function **phpinfo();**

phpinfo()

```php
<?php

phpinfo();
```



Test phpinfo (2014-03-26 13:55:51) - Log

Time	Severity	Details
26/03/2014 13:5...	output	phpinfo() PHP Version => 5.3.2-1ubuntu4.5
phpinfo() PHP Version => 5.3.2-1ubuntu4.5		
26/03/2014 13:5...	output	System => Linux jail01a 2.6.32-26-server #48-Ubuntu SMP Wed Nov 24 10:28:32
26/03/2014 13:5...	output	PHP API => 20090626 PHP Extension => 20090626 Zend Extension => 220090626 Zend Extension Build => A...
26/03/2014 13:5...	output	This server is protected with the Suhosin Patch 0.9.9.1 Copyright (c) 2006-2007 Hardened-PHP Project Copyri...
26/03/2014 13:5...	output	This program makes use of the Zend Scripting Language Engine: Zend Engine v2.3.0, Copyright (c) 1998-2010...
26/03/2014 13:5...	output	____
26/03/2014 13:5...	output	Configuration
26/03/2014 13:5...	output	bcmath
26/03/2014 13:5...	output	BCMath support => enabled
26/03/2014 13:5...	output	Directive => Local Value => Master Value bcmath.scale => 0 => 0
26/03/2014 13:5...	output	bz2
26/03/2014 13:5...	output	BZip2 Support => Enabled Stream Wrapper support => compress.bz2:// Stream Filter support => bzip2.decomp...
26/03/2014 13:5...	output	calendar
26/03/2014 13:5...	output	Calendar support => enabled
26/03/2014 13:5...	output	Core
26/03/2014 13:5	output	PHP Version => 5.3.2-1ubuntu4.5

PHP versions and configuration will change from time to time.

We endeavour to keep things backwards-compatible.

Deprecation warnings are enabled.

Php in 30 minutes (ish)

- For experienced programmers…

PHP Tags

- Your PHP code will be executed in Workbooks via a Web Server. Your code needs to differentiated from other elements that may occur within a Wen page.
- This is achieved by wrapping the PHP code within PHP tags, the most commonly used being:

  <?php…?>

Comments

- Single Line Comments

```
# This is a comment
```

- Multi - Line comments (Like C/C++)

```
/* This is a
     Multi-line comment
  */
```

Notes

- PHP statements end with a semi-colon.
- PHP is case sensitive
- PHP is whitespace insensitive
- Blocks of code are delimited with braces { …. }
- PHP supports functional and Object Oriented programming

Workbooks.com

Variables

- Denoted with a leading dollar sign.

```
$varname = 123;
```

- Variables are not typed when declared
- They can be used before they are assigned and will take on default values
- They are converted between types as required

Types

- Integer

```
$counter = 0;
```

- Double

```
$amount = 2.85
```

- Boolean

```
$account_active = TRUE;
```

- NULL

```
$var1 = NULL;
IsSet( $var1 ) will return FALSE
```

Types

- Strings

```
$message = “Hello World”;
```

  - Strings enclosed by double quotes support variable expansion. Single quotes do not.
  - PHP Supports Here Documents

- Arrays

```
$week_days = array( ‘Mon’, ‘Tue’, ‘Wed’,’Thu’, ‘Fri’ );
$days[0]  = ‘Sun’;
echo “The second working day is {$week_days[1]}”;
```

Types

- Associative Arrays (Hashes)

```
$classes = array(
   'Person' => 'Private::Crm::Person',
   'Lead' => 'Private::Crm::SalesLead',
);

echo 'Person class name is '.$classes['Person'];
```

- Tip: You can leave trailing commas on the last array entry

Four Variable Scopes

- Local - accessible within a function or module

```
$localVar = ‘abc’;
```

- Function Parameter - local within function

- Global - within or without any function

```
GLOBAL $varname;
```

- Static (like C/C++)
  - Will retain value over multiple functions calls

```
STATIC $call_count;
```

Constants

```
define( LINE_LENGTH, 255 );
echo LINE_LENGTH;  /* Note No leading $ */
```

- Magic Constants (See http://www.php.net/manual/en/language.constants.predefined.php)

```
   __LINE__
   __FILE__
   __DIR__
   __FUNCTION__
```

Conditionals

```
if ( $line_length < LINE_LIMIT )
{
    echo ‘Line length is within limits’;
}
else
{
    echo ‘Line length exceeds limit: ‘.LIMIT;
}
```

Loops

```
while ( expression )
{
   /* block of code */
}


do
{
   /* block of code */
} while ( expression )


for ( expression1; expression2; expression3 )
{
   /* block of code */
}
```

Loops

- foreach - iterates over an array

```
foreach ( $array as $value )
{
   /* Block of code */
}
```

- Or iterate over the members of a hash:

```
foreach ($hash as $key => $value)
{
 /* Block of code */
}
```

Functions

```
function myFunction( $param1 = “No Value”, $param2  = “Value” )
{
    return “$param1 $param2”;
}
```

- Parameters are passed by value. Prefix with & to pass by reference.

Classes

```
class WorkBooksConnection extends Connections
{
   /* Variable Declarations */
   /* Function Declarations */
}
```

Use:

```
$wb = new WorkBooksConnection;
$wb->publicMethod1( $param1 );
```

Class Members

```
# Constants are declared with the const keyword
const REQUEST_SIZE = 1024;
# A public attribute
var $var1 = 123;
# A private instance variable, only accessible by functions within this class.
private $var2 = 456;
# A private instance variable, accessible by any sub-class
protected $var3 = 789;
```

protected and **private** may also be used to set the scope for member functions too.

Constructors and Destuctors

- A constructor function may be declared:

```
function __contruct( $param1, $param2 ) { …. }
```

- A destructor may function may also be declared:

function __destruct() { …. }

Many Available Functions and Classes

- PHP has many many functions and classes available to the programmer.
  - String functions
  - Web Services
  - JSON parsing
  - Email handling
  - Exception handling
  - Many Many More.

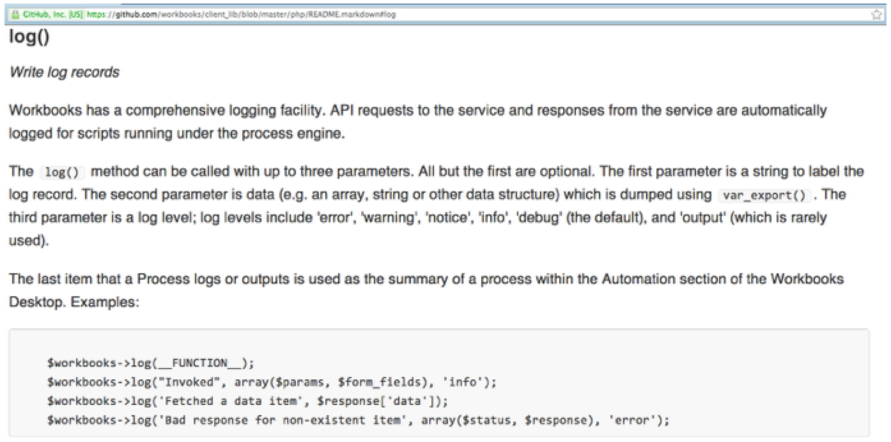
http://www.php.net/manual/en/funcref.php

More Information

- Google
- http://www.php.net/docs.php

Logging

- API calls log automatically.
- Use $workbooks->log() often.
- Last line treated as the “summary”.
- It’s recommended that you log the inputs to your process


GitHub, Inc. [US] https://github.com/workbooks/client_lib/blob/master/php/README.markdown#log

log()

Write log records

Workbooks has a comprehensive logging facility. API requests to the service and responses from the service are automatically logged for scripts running under the process engine.

The `log()` method can be called with up to three parameters. All but the first are optional. The first parameter is a string to label the log record. The second parameter is data (e.g. an array, string or other data structure) which is dumped using `var_export()`. The third parameter is a log level; log levels include 'error', 'warning', 'notice', 'info', 'debug' (the default), and 'output' (which is rarely used).

The last item that a Process logs or outputs is used as the summary of a process within the Automation section of the Workbooks Desktop. Examples:

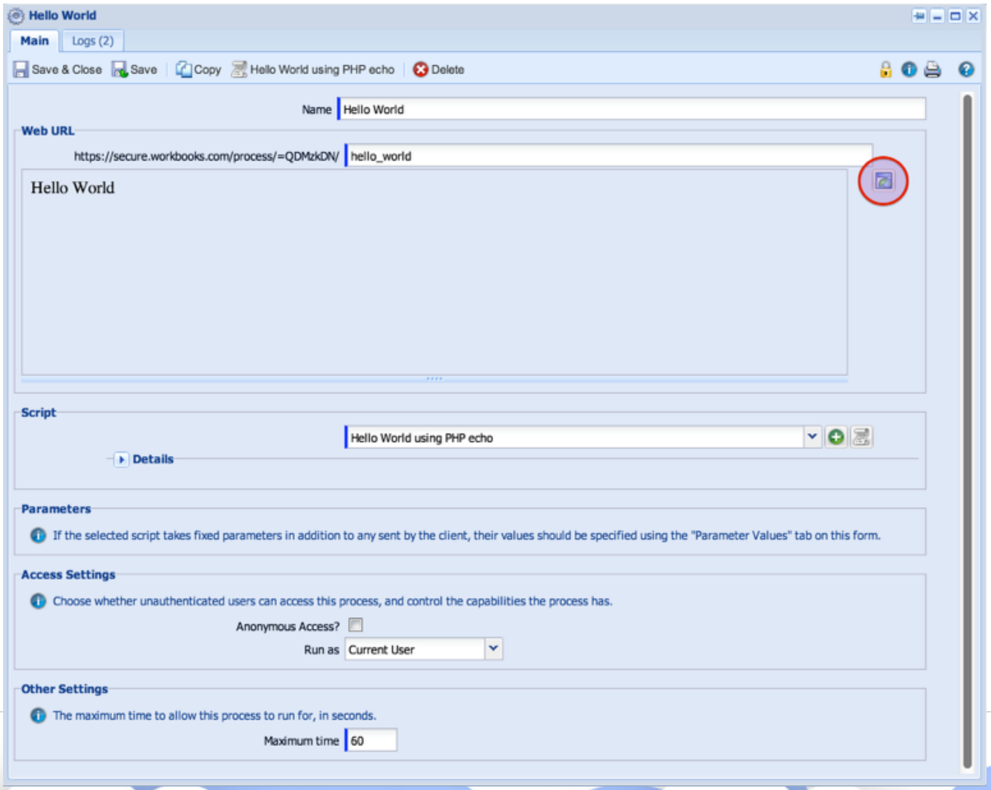
```
$workbooks->log(__FUNCTION__);
$workbooks->log("Invoked", array($params, $form_fields), 'info');
$workbooks->log('Fetched a data item', $response['data']);
$workbooks->log('Bad response for non-existent item', array($status, $response), 'error');
```

```
<?php

$workbooks->log("Hello World");

// $workbooks->log() takes one or more parameters.
$workbooks->log('Process inputs',
  array( // The second parameter, if present, is passed through var_export()
    '$_POST' => $_POST,
    '$_GET' => $_GET,
    '$_SERVER' => $_SERVER
  )
);

$workbooks->log('Process inputs',
  array(
    '$_POST' => $_POST,
    '$_GET' => $_GET,
    '$_SERVER' => $_SERVER
  ),
  'info', // Log level (default: 'debug')
  1000000 // Maximum log length (default: 4096 bytes)
);
```

Web Processes

- Process output shown in preview 'iframe' on form, logs in tab
  - Or click on button circled to open in another window and capture URL
  - URL identifies database
- Access Settings
  - Can be used to permit anonymous access
- Headers
  - Set headers prior to any output
  - Use $workbooks->header()

    e.g. to set cookies
- Output streamed to client as it is sent by the script
- Output UTF-8, be sure to escape it: use ***htmlentities()***



Name: Hello World

Web URL

https://secure.workbooks.com/process/=QDMzkDN/ hello_world

Hello World

Script: Hello World using PHP echo

Parameters

If the selected script takes fixed parameters in addition to any sent by the client, their values should be specified using the "Parameter Values" tab on this form.

Access Settings

Choose whether unauthenticated users can access this process, and control the capabilities the process has.

Anonymous Access? ☐

Run as: Current User

Other Settings

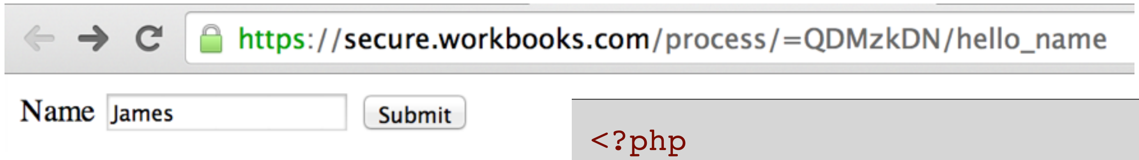
The maximum time to allow this process to run for, in seconds.

Maximum time: 60

Exercise: Hello “name”

- The challenge: show an HTML form asking for a name using a script hosted within the Process Engine.
- Use the ‘Web Process’ facility.
- Echo that name back.
- Hint: URL parameters are put in **$_GET[]** and **$_POST[].**

Example: Hello "name"


https://secure.workbooks.com/process/=QDMzkDN/hello_name

Name James Submit

```
<?php
if (empty($_POST)) {
  echo <<<EOF
    <form method="post">
      <label for="name">Name</label>
      <input name="name"/>
      <input type="submit" value="Submit"/>
    </form>
EOF;
}
else {
  $encoded_name = htmlentities($_POST['name']);
  echo "<p>Hello {$encoded_name}</p>";
}
```

Parameters

As per ‘standard’ PHP norms, the Process Engine sets these up:

- **URL** parameters are put in **$_GET[]** and **$_POST[]**
- **Environment** parameters are in **$_SERVER[]**
- Uploaded **File** parameters are in **$_FILE[]**

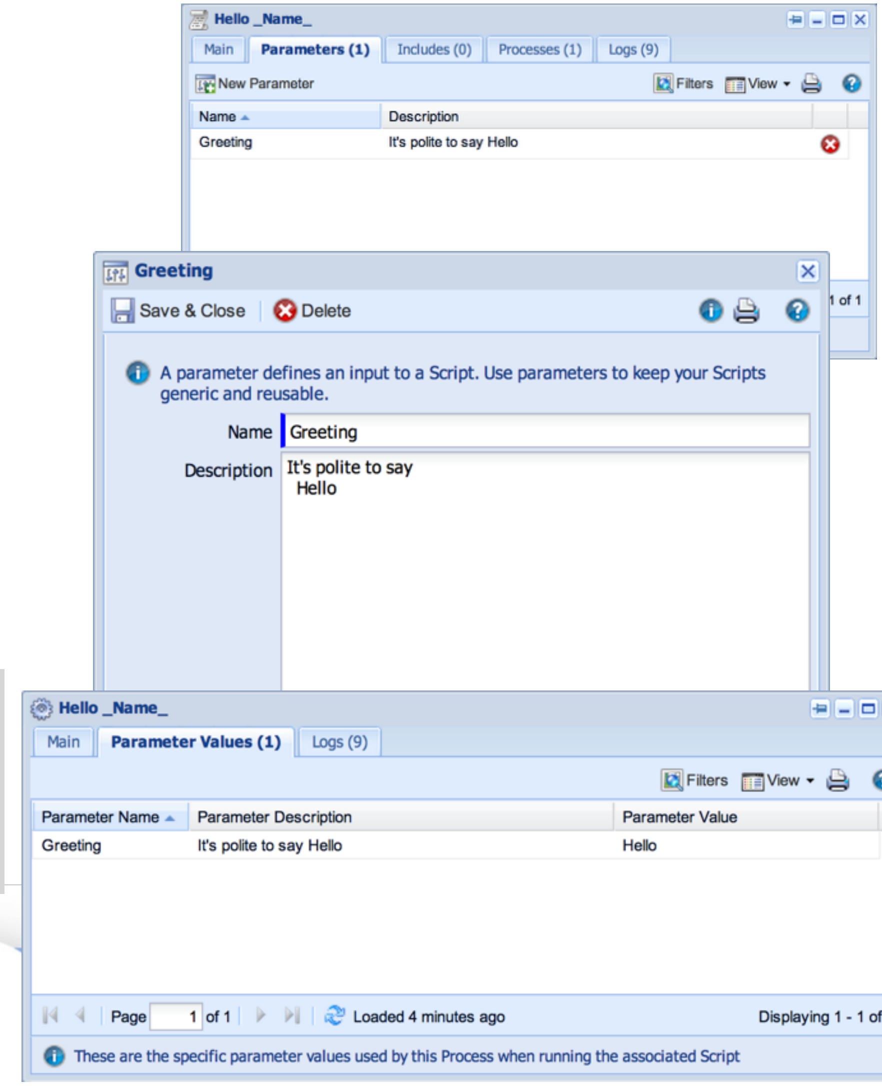
In addition these are also set up:

- **‘Script’** parameters are in **$params[]**
- **‘Form field’** parameters are in **$form_fields[]**
- **$workbooks**

Using Script Parameters

- On your Script go to the Parameters tab, give it a name and description.
- Open a Process which calls the Script, define a value.
- That value will be passed to the script within the **$params[]** array.
- Common pattern: several processes invoke a script with differing parameters
- In the previous example…

```
echo "<p>{$params['Greeting']}
{$encoded_name}</p>";
```



Name	Description
Greeting	It's polite to say Hello

A parameter defines an input to a Script. Use parameters to keep your Scripts generic and reusable.

Name: Greeting

Description: It's polite to say
Hello

Parameter Name	Parameter Description	Parameter Value
Greeting	It's polite to say Hello	Hello

Page 1 of 1 Loaded 4 minutes ago Displaying 1 - 1 of 1

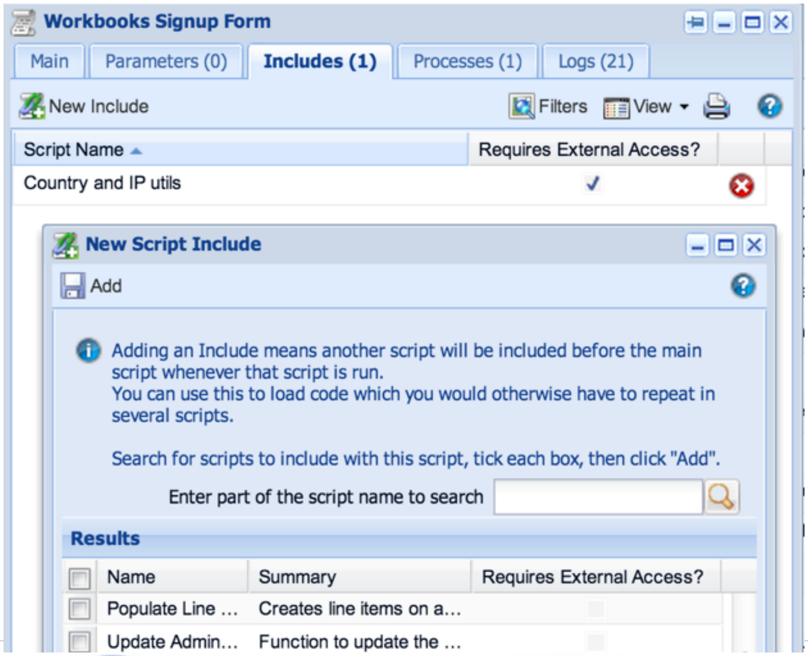
These are the specific parameter values used by this Process when running the associated Script

Exercise: Add script parameters to ‘Hello Name’

- Adapt your previous code to use Script Parameters

Including shared code

- A simple mechanism to share code between scripts
- Usual model: common functions and constants



Script Name	Requires External Access?
Country and IP utils	✓

Adding an Include means another script will be included before the main script whenever that script is run.
You can use this to load code which you would otherwise have to repeat in several scripts.

Search for scripts to include with this script, tick each box, then click "Add".

Enter part of the script name to search

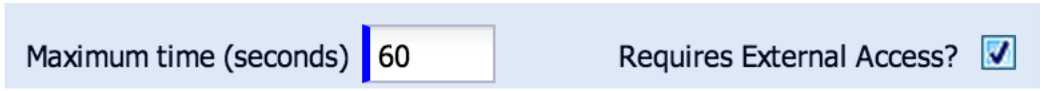
	Name	Summary	Requires External Access?
☐	Populate Line …	Creates line items on a…	
☐	Update Admin…	Function to update the …	

Workbooks.com

How the Jail runs processes

- Processes are run within a sandbox, away from the main Workbooks service.
- Passed to the process each time it's run:
  - Scripts, included scripts, parameters, HTTP parameters, environment variables
  - One special script: workbooks_api.php
  - Each included script is run in turn, with the main script run last.
- Processes only have write access to their TMPDIR.
- Processes authenticate automatically back to Workbooks using credentials passed when they are invoked.

Jail Resource Constraints


Maximum time (seconds) 60

Requires External Access? ✓

- An alarm timer limits the process to its allocated maximum time.
  - Discover the maximum time in seconds using **getenv('TIMEOUT')**
- If ‘Requires External Access’ is set, firewall ruleset is more open:
  - ICMP
  - DNS
  - HTTP, HTTPS
  - IMAP, IMAPS, POP3, POP3S
  - Database (MySQL, MSSQL default ports)
- Specifically not SMTP: use Workbooks’ email API instead.
- Memory, disk usage, process limits all enforced
  - Receive a SIGTERM if memory limit exceeded.
- Workbooks recommends that processes do their work in small batches and checkpoint if required.

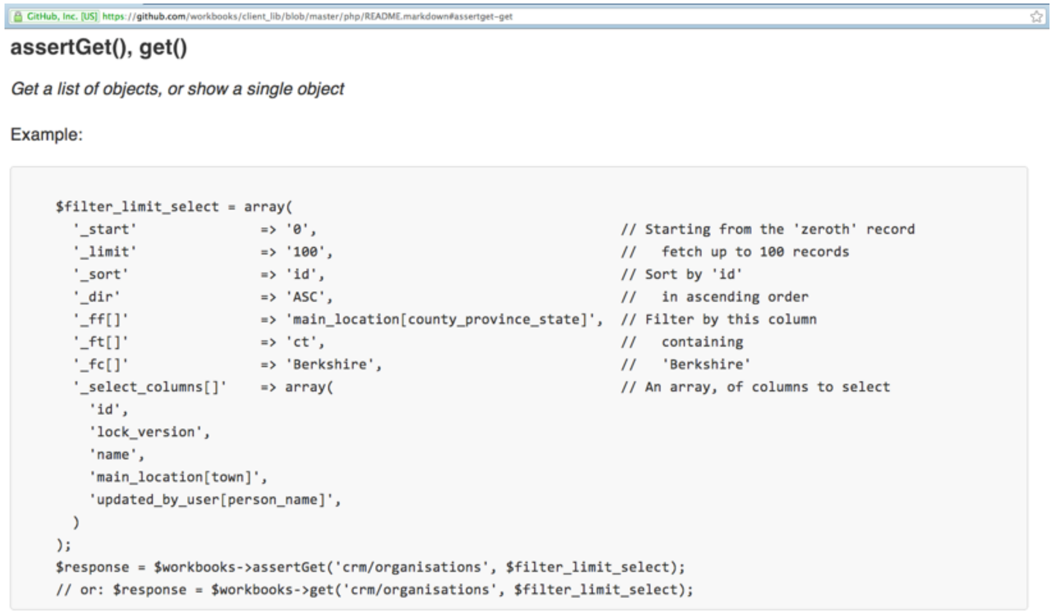
Accessing the Service

- Within the Process Engine:
  - Use $workbooks->assertGet(), assertCreate(), assertUpdate(), assertDelete()
  - Authentication and logout is automatic
- Using the wire protocol:
  - HTTP POST or GET, be careful with URL encoding especially with sparse arrays
  - Explicitly authenticate before you start
  - https://secure.workbooks.com/…
  - Do not turn off certificate checks.
  - Test using ‘curl’, as per examples in API Developer Guide

Workbooks.com

Using the API: Get records

- Retrieve Parameters: all optional:
  - start, limit (default: 100)
  - sort, direction
  - filter
  - column selection (speed)
- Response:
  - an array of hashes
- Errors:
  - **<u>assert</u>**Get - raise exception



GitHub, Inc. [US] https://github.com/workbooks/client_lib/blob/master/php/README.markdown#assertget-get

assertGet(), get()

Get a list of objects, or show a single object

Example:

```
$filter_limit_select = array(
  '_start'               => '0',                                      // Starting from the 'zeroth' record
  '_limit'               => '100',                                    //   fetch up to 100 records
  '_sort'                => 'id',                                     // Sort by 'id'
  '_dir'                 => 'ASC',                                    //   in ascending order
  '_ff[]'                => 'main_location[county_province_state]',   // Filter by this column
  '_ft[]'                => 'ct',                                     //   containing
  '_fc[]'                => 'Berkshire',                              //   'Berkshire'
  '_select_columns[]'    => array(                                    // An array, of columns to select
    'id',
    'lock_version',
    'name',
    'main_location[town]',
    'updated_by_user[person_name]',
  )
);
$response = $workbooks->assertGet('crm/organisations', $filter_limit_select);
// or: $response = $workbooks->get('crm/organisations', $filter_limit_select);
```

More about fetching data

- ‘total’ element: the total number of matching rows
  - Fetch with _limit set to 0 to discover this without retrieving the rows (remember to set _start as well)
  - Set **_skip_total_rows=1** (or true) to speed things up
- Most records have common fields such as id, lock_version, object_ref, name.
  - Tip: object_ref is convenient in your logging since you can paste it into the search bar
- PDF, and ‘.print’ versions. ‘csv’ version.
- Report
  - A good way to wrap up a lot of complexity, away from code
- Metadata API
  - Discover the set of fields, including custom fields
- Do not assume field order or record order without sort
  - **id** is assigned in ascending order
- Field sizes are important

Filters

- In general: sets of {field, comparison operator, value}
- Combine with boolean logic (defaults to just ‘AND’)
- OR can be slow: avoid this when many records queried

- Several syntaxes - see https://github.com/workbooks/client_lib/blob/master/php/filter_example.php

- Choose whichever suits your needs

Filter syntax 1: arrays of fields, comparators, contents

```
// First filter structure: specify arrays for Fields ('_ff[]'), comparaTors ('_ft[]'), Contents ('_fc[]').
// Note that 'ct' (contains) is MUCH slower than equals. 'not_blank' requires Contents to compare with, but this is ignored.
$filter1 = array_merge($limit_select, array(
  '_ff[]' => array('main_location[county_province_state]', ‘main_location[county_province_state]', 'main_location[street_address]'),
  '_ft[]' => array('eq',                                   'ct',                                    'not_blank'),
  '_fc[]' => array('Berkshire',                            'Yorkshire',                              ''),
  '_fm' => '(1 OR 2) AND 3',                            // How to combine the above clauses, without this: 'AND'.
));
$response1 = $workbooks->assertGet('crm/organisations', $filter1);
$workbooks->log('Fetched objects using filter1', array($filter1, $response1['data']));
```

Filter syntax 2: JSON-formatted

```
// The equivalent using a second filter structure: a JSON-formatted string  array of arrays containg 'field, comparator, contents'
$filter2 = array_merge($limit_select, array(
  '_filter_json' => '['.
    '["main_location[county_province_state]", "eq", "Berkshire"],' .
    '["main_location[county_province_state]", "ct", "Yorkshire"],' .
    '["main_location[street_address]", "not_blank", ""]' .
    ']',
  '_fm' => '(1 OR 2) AND 3',                             // How to combine the above clauses, without this: 'AND'.
));
$response2 = $workbooks->assertGet('crm/organisations', $filter2);
$workbooks->log('Fetched objects using filter2', array($filter2, $response2['data']));
```

Filter syntax 3: array of filters

```
// The equivalent using a third filter structure: an array of filters, each containg 'field, comparator, contents'.
$filter3 = array_merge($limit_select, array(
  '_filters[]'     => array(
     array('main_location[county_province_state]', 'eq', 'Berkshire'),
     array('main_location[county_province_state]', 'ct', 'Yorkshire'),
     array('main_location[street_address]', 'not_blank', ''),
  ),
  '_fm' => '(1 OR 2) AND 3',                            // How to combine the above clauses, without this: 'AND'.
));
$response3 = $workbooks->assertGet('crm/organisations', $filter3);
$workbooks->log('Fetched objects using filter3', array($filter3, $response3['data']));
```

Filter comparison operators

- Many, see the API Developers Guide for a full list
- **eq** - equality, **ne** - not equals
- **bg** - begins with, **nbg** - does not begin with
- **ct** - contains, **nct** - does not contain
- Some do not require a value - specify '':
  - **false**, **true**, **blank**, **not_blank**, **today**
- **eq** and **bg** much faster than **ct** etc.

Using a report with filters

- Benefit: combine data from several related records
- Apply filters to select specific record(s)
- Report definition encapsulates complex rules
    - Don’t try to be too complex (performance)
    - Consider indexes
- Fetch report metadata to discover columns etc
- See report_examples.php

```
$view_name='Inventory:By Competition';
$escaped_data_view_name = rawurlencode($view_name);
$response = $workbooks->assertGet("data_view/{$escaped_data_view_name}", $filter);
```

Exercise: Fetch people records

- Write a script to fetch all People whose county is ‘Berkshire’.

Create

- Batch
  - up to 100 objects in a single request (which can become slow: consider smaller batches, maybe 10 or 20)
- Wire proto: id=0, lock_version=0
- Response: an array of
  - ‘affected objects’ - id, lock_version, …
- Create:
  - main objects
  - picklists and associations
  - dynamic linked items
  - relationships

Workbooks.com

Create a record

Simple create…

```
$create_one_organisation = array(
  'name'                                 => 'Birkbeck Burgers',
  'industry'                             => 'Food',
  'main_location[country]'               => 'United Kingdom',
  'main_location[county_province_state]' => 'Oxfordshire',
  'main_location[town]'                  => 'Oxford',
);
$response = $workbooks->assertCreate(
  'crm/organisations', $create_one_organisation);
$created_id_lock_versions = $workbooks->idVersions($response);
```

$response contains 'affected_objects' hash (id, lock_version, …)

Picklists

- Workbooks picklists are simple lists of string values
  - Some picklists have ‘open/closed’ state
  - ‘Resricted’ or ‘Unrestricted’
- Two endpoints:
  - admin/picklists: picklists
    (id, name)
  - admin/picklists: picklist_entries
    (picklist_id, value, display_order)

Workbooks.com

Fetch picklist entries

```
/**
* Fetch the contents of a picklist, caching the response
*/
private function fetch_picklist_entries_and_cache($picklist_id) {
  static $result_cache = array();

  if (!isset($result_cache[$picklist_id])) {
    $picklist_api = 'picklist_data/Private_PicklistEntry/id/value';
    $response = $this->workbooks->assertGet($picklist_api,
       array('picklist_id' => $picklist_id));
    $result_cache[$picklist_id] = @$response;
  }
  return $result_cache[$picklist_id];
} // fetch_picklist_entries_and_cache()
```

Associations

- Associated records are linked by ID.
- Entries on a queue
- e.g. relationship to ‘parent’
  - Campaign member: marketing_campaign_id
  - Specify IDs when modifying
- relationships between records
- Separately: relationship APIs for many:many

Relationship APIs

- API endpoint depends on the record type
    - They differ due to additional fields such as relationship statuses

- **accounting/document_header_relationships**
    - between ‘transaction documents’
- **accounting/document_header_contacts**
    - between ‘transaction documents’ and ‘parties’
- **activity/activity_links**
    - between ‘activities’ and other items
- **related_items**
    - everything else

Create relationships

```
$create_relationships = array(
        'source_id' => $form_fields['id'],
        'source_type' => $form_fields['type'],
        'related_item_id' => $line_item[$all_fields_field],
        'related_item_type' => 'Private::Crm::MarketingCampaign',
);
$workbooks->assertCreate('related_items', $create_relationships);
```

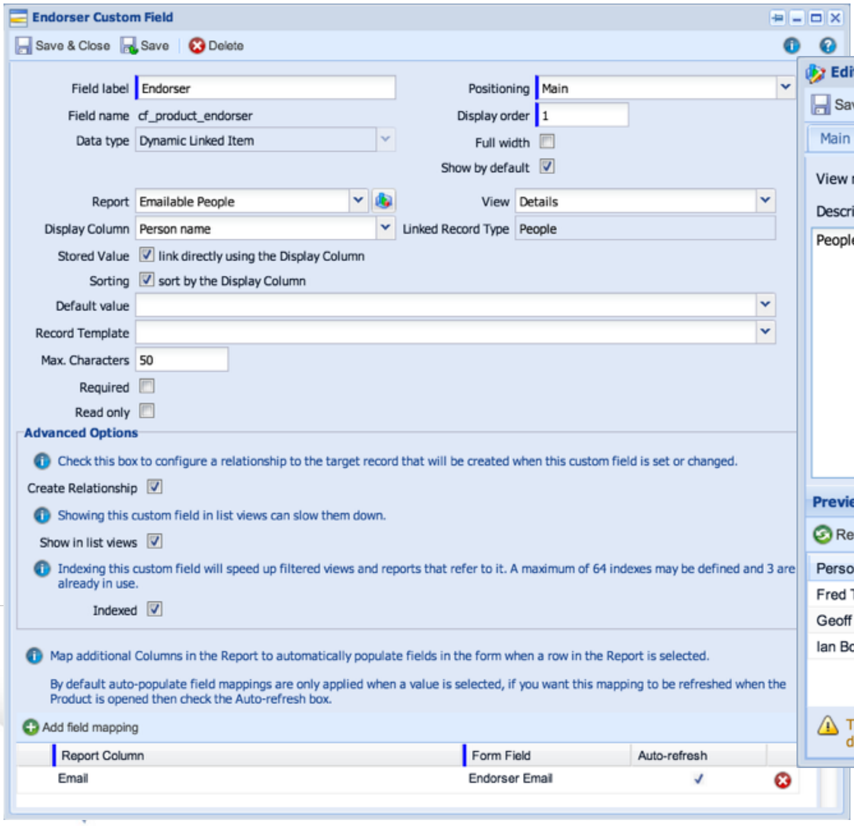
Relationships are normally bi-directional.

Dynamic Linked Items

- Dynamic Linked Items (DLIs)
- DLIs are custom associations between records
  - Populated through a report
  - Assign an ID to: **linked_item_association_for_***cfname*
    - where cfname is the target field name
  - The report used in a DLI must yield an ID field and typically has a Name field for display purposes

DLI Configuration example

- Field: cf_product_endorser
- Set using linked_item_association_for_cf_product_endorser
- In this case, would be the ID of a Person



Endorser Custom Field

Save & Close | Save | Delete

Field label	Endorser	Positioning	Main
Field name	cf_product_endorser	Display order	1
Data type	Dynamic Linked Item	Full width	☐
		Show by default	✓
Report	Emailable People	View	Details
Display Column	Person name	Linked Record Type	People
Stored Value	✓ link directly using the Display Column		
Sorting	✓ sort by the Display Column		
Default value			
Record Template			
Max. Characters	50		
Required	☐		
Read only	☐		

Advanced Options

Check this box to configure a relationship to the target record that will be created when this custom field is set or changed.

Create Relationship ✓

Showing this custom field in list views can slow them down.

Show in list views ✓

Indexing this custom field will speed up filtered views and reports that refer to it. A maximum of 64 indexes may be defined and 3 are already in use.

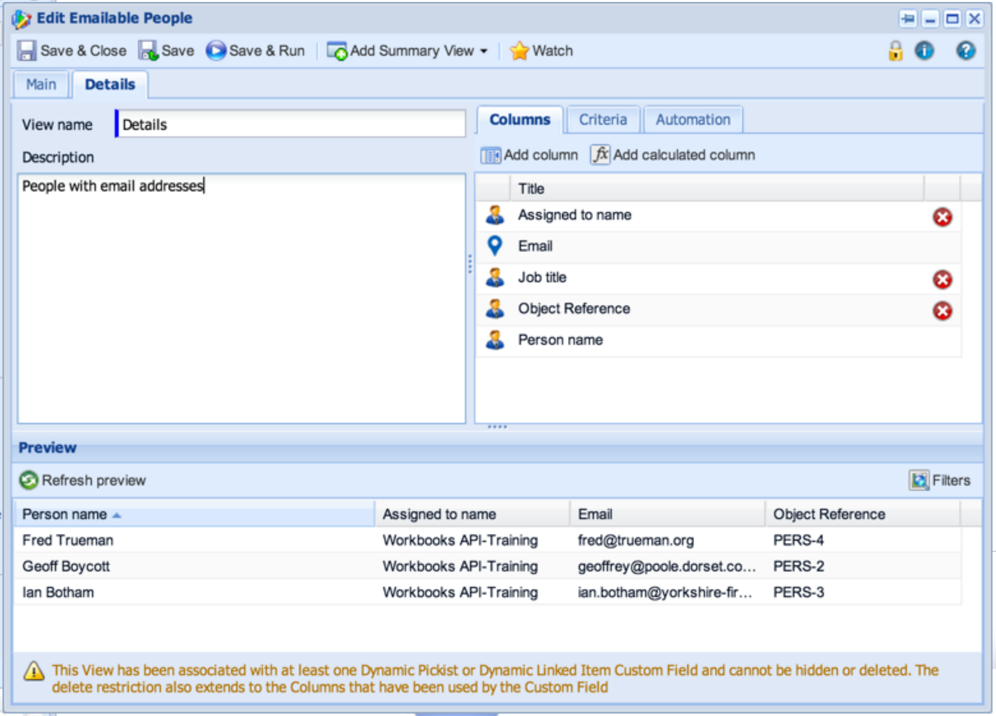
Indexed ✓

Map additional Columns in the Report to automatically populate fields in the form when a row in the Report is selected.

By default auto-populate field mappings are only applied when a value is selected, if you want this mapping to be refreshed when the Product is opened then check the Auto-refresh box.

Add field mapping

Report Column	Form Field	Auto-refresh
Email	Endorser Email	✓



Edit Emailable People

Save & Close | Save | Save & Run | Add Summary View | Watch

Main | **Details**

View name: Details

Description: People with email addresses

Columns | Criteria | Automation

Add column | Add calculated column

Title
Assigned to name
Email
Job title
Object Reference
Person name

Preview

Refresh preview | Filters

Person name	Assigned to name	Email	Object Reference
Fred Trueman	Workbooks API-Training	fred@trueman.org	PERS-4
Geoff Boycott	Workbooks API-Training	geoffrey@poole.dorset.co...	PERS-2
Ian Botham	Workbooks API-Training	ian.botham@yorkshire-fir...	PERS-3

This View has been associated with at least one Dynamic Picklist or Dynamic Linked Item Custom Field and cannot be hidden or deleted. The delete restriction also extends to the Columns that have been used by the Custom Field

Queues

- Queues are collections of records
  - Simple workflow through assignment
- Records are assigned to Queues
- Users subscribe to Queues
- Each Queue **has separate IDs**

  e.g. The assigned_to value for a person record is not compatible with that for an task

Queue Query - by ID

```
/**
* Returns a queue name for a given queue ID and object type (specify the API), caching the result for efficiency.
*/
private function get_queue_name($queue_id, $api) {
  static $result_cache = array();

  if (empty($queue_id)) { return NULL; }
  $cache_key = "{$api}:{$queue_id}";
  if (!isset($result_cache[$cache_key])) {
    $get_queue = array(
      '_ff[]' => array('id'),
      '_ft[]' => array('eq'),
      '_fc[]' => array($queue_id),
      '_select_columns[]' => array('name'),
    );
    $response = $this->workbooks->assertGet($api, $get_queue);
    $result_cache[$cache_key] = @$response['data'][0]['name'];
  }
  return $result_cache[$cache_key];
} // get_queue_name()
private function get_person_queue_name($queue_id){ return $this->get_queue_name($queue_id, 'crm/person_queues'); }
private function get_sales_lead_queue_name($queue_id){ return $this->get_queue_name($queue_id, 'crm/sales_lead_queues'); }
```

Queue Query - by Name

```
/**
* Returns a queue ID for a given queue name and object type (specify the API), caching the result for efficiency.
*/
private function get_queue_id($queue_name, $api) {
  static $result_cache = array();

  if (empty($queue_name)) { return NULL; }
  $cache_key = "{$api}:{$queue_name}";
  if (!isset($result_cache[$cache_key])) {
    $get_queue = array(
      '_ff[]' => array('name'),
      '_ft[]' => array('eq'),
      '_fc[]' => array($queue_name),
      '_select_columns[]' => array('id'),
    );
    $response = $this->workbooks->assertGet($api, $get_queue);
    $result_cache[$cache_key] = @$response['data'][0]['id'];
  }
  return $result_cache[$cache_key];
} // get_queue_id()
private function get_activity_queue_id($queue_name){ return $this->get_queue_id($queue_name, 'activity/activity_queues'); }
private function get_person_queue_id($queue_name){ return $this->get_queue_id($queue_name, 'crm/person_queues'); }
private function get_sales_lead_queue_id($queue_name){ return $this->get_queue_id($queue_name, 'crm/sales_lead_queues'); }
```

Queue Query - by Name

So to access the queue to which a record is assigned…

```
  /**
  *  Returns the queue name which the given record is assigned to
  */
  private function get_assigned_queue_name($record){
    switch($record['type']) {
      case 'Private::Crm::Person': return $this->get_person_queue_name($record['assigned_to']);
      case 'Private::Crm::SalesLead': return $this->get_sales_lead_queue_name($record[‘assigned_to']);
//// etc
      return NULL;
    }
  } // get_assigned_queue_name()
```

Data types

- Currency values:
  - *code amount fix* e.g. GBP 123.45 0
  - code: 3-character ISO-4217 code (GBP USD EUR JPY CHF)
  - fix: ignore this, set to zero (both currency and code can change)
- Dates, DateTimes - use the ‘C’ locale
  - "due_date" : "22 May 2009”
    - output format: %e %b %Y
  - "updated_at" : "Fri May 15 14:36:54 UTC 2009”
    - output format: %a %b %d %H:%M:%S %Z %Y
  - internally stored as seconds since the epoch
  - input formats are a little more flexible than this

Another ‘create’ example: Quote

```
$today = date('Y-m-d');
$create_quote = array(
  'party_id'       => $party_id, //Set the party ID (Customer)
  'description'    => 'New quotation',
  'document_date'    => $today,
  'document_currency' => 'GBP',
);

$creation_response = $workbooks->assertCreate('accounting/quotations', $create_quote);
// $creation_response[‘affected_objects’] is an array of hashes, each containing
// a number of fields including ‘id’ and ‘lock_version’:
//
//    @$creation_response[‘affected_objects’][0][‘id’]
//    @$creation_response[‘affected_objects’][0][‘lock_version’]
```

Line Items

Workbooks.com

- Creating Line Items, you must specify the ID of the item that they are linked to (document_header_id)
- e.g. for a Contract:

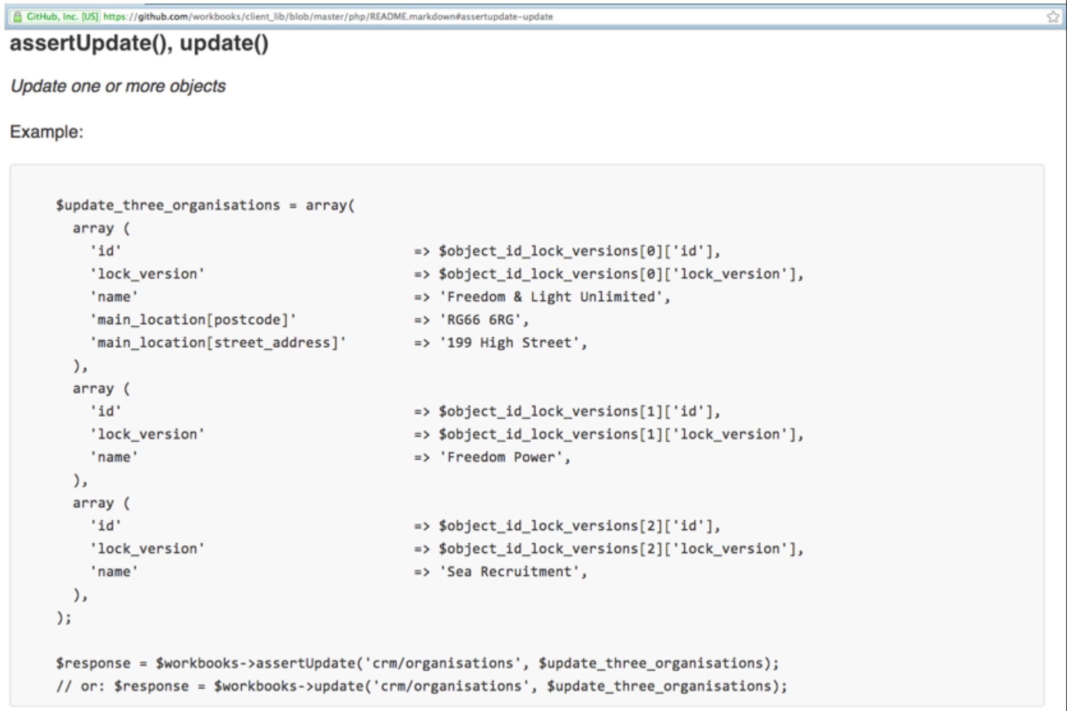
```
function add_interval_to_date($date_str, $interval='P1Y', $format='Y-m-d') { // Add interval to
date_str, returning date in format. Date must be in parseable format.
  $d = new DateTime($date_str);
  $d->add(new DateInterval($interval));
  return $d->format($format);
} // add_interval_to_date()

$create_contract_line_item = array(
    'document_header_id' => $last_contract_line_item['document_header_id'],
    'end_date' => add_interval_to_date($last_contract_line_item['end_date'], $extend_by),
    'start_date' => add_interval_to_date($last_contract_line_item['end_date'], 'P1D'),
    'description' => $last_contract_line_item['description'],
    'product_id' => $last_contract_line_item['product_id'],
    'unit_quantity' => $last_contract_line_item['unit_quantity'],
    'document_currency_unit_price_value' =>
$last_contract_line_item['document_currency_unit_price_value'],
  );
  $workbooks->log('About to create a new contract line item', $create_contract_line_item);
  $workbooks->assertCreate('accounting/contract_line_items', $create_contract_line_item);
```

PHP's DateInterval class is useful for adding and subtracting time periods.

Logging the parameters to your assertCreate() results in better log information than relying on automatic logging.

Update

- Required:
  - id
  - lock_version
  - _can_modify capability
  - fields to change
- Returns
  - An array of affected objects/errors



GitHub, Inc. [US] https://github.com/workbooks/client_lib/blob/master/php/README.markdown#assertupdate-update

assertUpdate(), update()

Update one or more objects

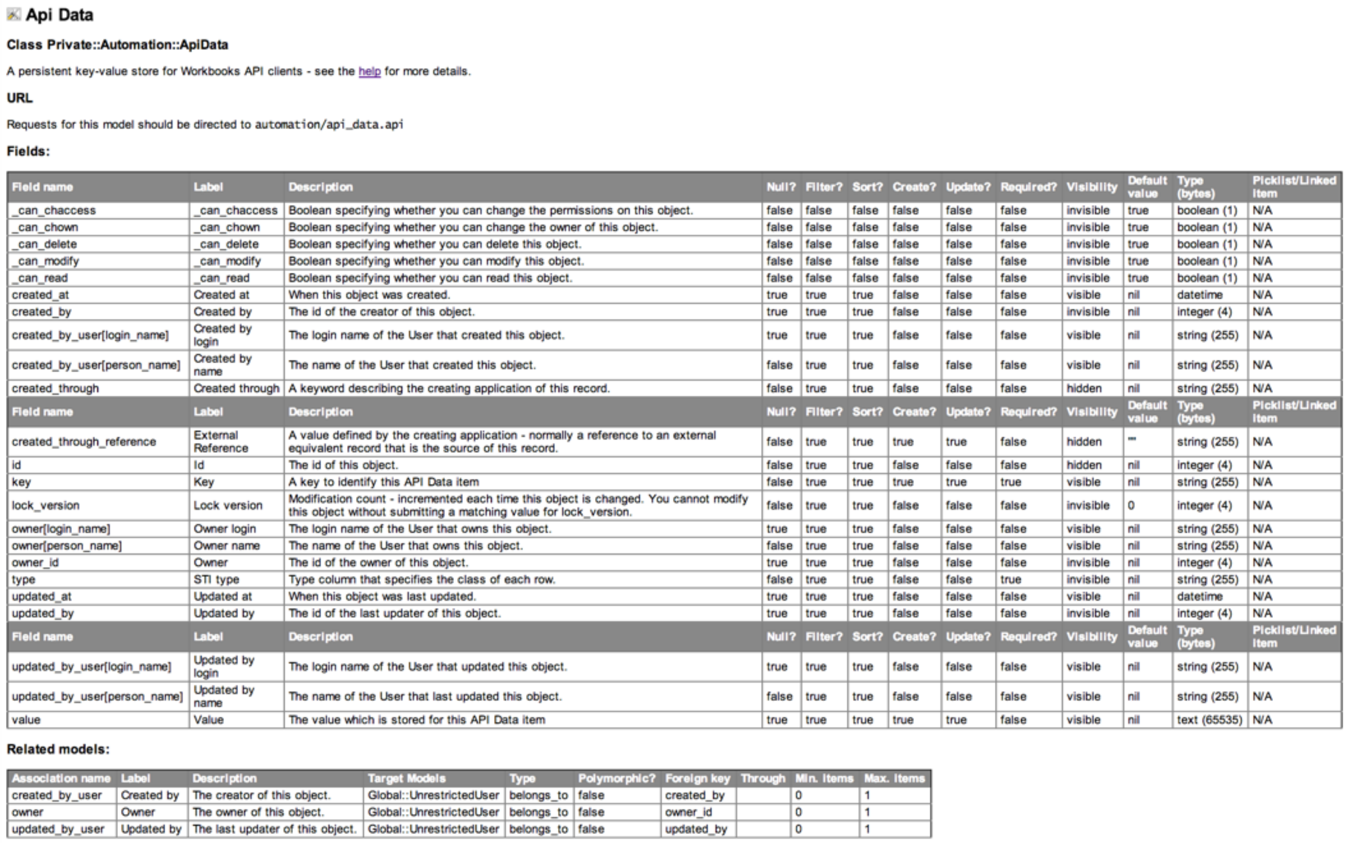
Example:

```
$update_three_organisations = array(
  array (
    'id'                                => $object_id_lock_versions[0]['id'],
    'lock_version'                      => $object_id_lock_versions[0]['lock_version'],
    'name'                              => 'Freedom & Light Unlimited',
    'main_location[postcode]'           => 'RG66 6RG',
    'main_location[street_address]'     => '199 High Street',
  ),
  array (
    'id'                                => $object_id_lock_versions[1]['id'],
    'lock_version'                      => $object_id_lock_versions[1]['lock_version'],
    'name'                              => 'Freedom Power',
  ),
  array (
    'id'                                => $object_id_lock_versions[2]['id'],
    'lock_version'                      => $object_id_lock_versions[2]['lock_version'],
    'name'                              => 'Sea Recruitment',
  ),
);

$response = $workbooks->assertUpdate('crm/organisations', $update_three_organisations);
// or: $response = $workbooks->update('crm/organisations', $update_three_organisations);
```

- Stale object error:
  - lock_version out of date

Metadata

- Calls retrieve this for you - see metadata_example.php
- The documentation in the API Reference Guide is invaluable



Api Data

Class Private::Automation::ApiData

A persistent key-value store for Workbooks API clients - see the help for more details.

URL

Requests for this model should be directed to `automation/api_data.api`

Fields:

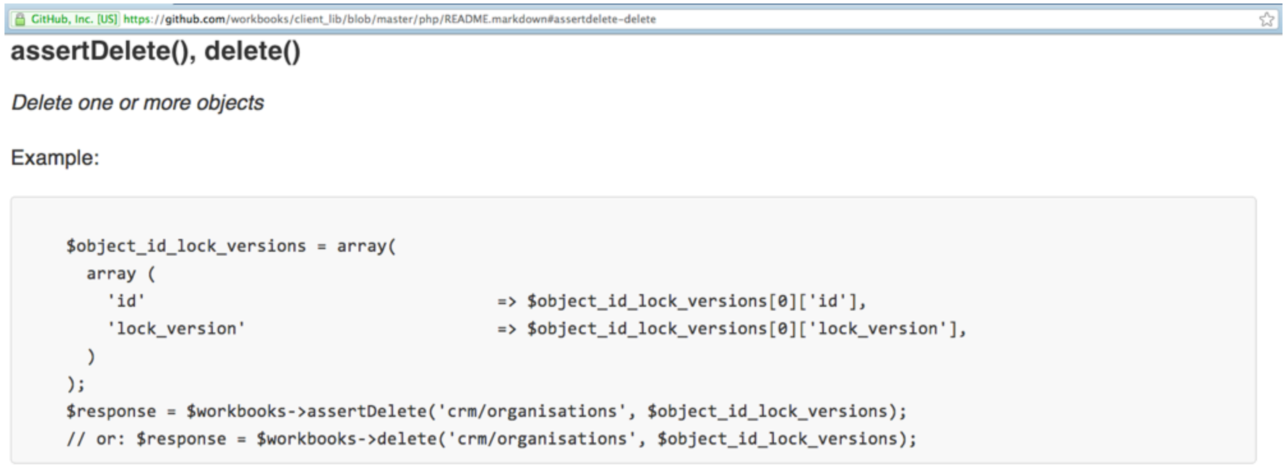
Field name	Label	Description	Null?	Filter?	Sort?	Create?	Update?	Required?	Visibility	Default value	Type (bytes)	Picklist/Linked item
_can_chaccess	_can_chaccess	Boolean specifying whether you can change the permissions on this object.	false	false	false	false	false	false	invisible	true	boolean (1)	N/A
_can_chown	_can_chown	Boolean specifying whether you can change the owner of this object.	false	false	false	false	false	false	invisible	true	boolean (1)	N/A
_can_delete	_can_delete	Boolean specifying whether you can delete this object.	false	false	false	false	false	false	invisible	true	boolean (1)	N/A
_can_modify	_can_modify	Boolean specifying whether you can modify this object.	false	false	false	false	false	false	invisible	true	boolean (1)	N/A
_can_read	_can_read	Boolean specifying whether you can read this object.	false	false	false	false	false	false	invisible	true	boolean (1)	N/A
created_at	Created at	When this object was created.	true	true	true	false	false	false	visible	nil	datetime	N/A
created_by	Created by	The id of the creator of this object.	true	true	true	false	false	false	invisible	nil	integer (4)	N/A
created_by_user[login_name]	Created by login	The login name of the User that created this object.	true	true	true	false	false	false	visible	nil	string (255)	N/A
created_by_user[person_name]	Created by name	The name of the User that created this object.	false	true	true	false	false	false	visible	nil	string (255)	N/A
created_through	Created through	A keyword describing the creating application of this record.	false	true	true	false	false	false	hidden	nil	string (255)	N/A
created_through_reference	External Reference	A value defined by the creating application - normally a reference to an external equivalent record that is the source of this record.	false	true	true	true	true	false	hidden	""	string (255)	N/A
id	Id	The id of this object.	false	true	true	false	false	false	hidden	nil	integer (4)	N/A
key	Key	A key to identify this API Data item	false	true	true	true	true	true	visible	nil	string (255)	N/A
lock_version	Lock version	Modification count - incremented each time this object is changed. You cannot modify this object without submitting a matching value for lock_version.	false	true	true	false	false	false	invisible	0	integer (4)	N/A
owner[login_name]	Owner login	The login name of the User that owns this object.	true	true	true	false	false	false	visible	nil	string (255)	N/A
owner[person_name]	Owner name	The name of the User that owns this object.	false	true	true	false	false	false	visible	nil	string (255)	N/A
owner_id	Owner	The id of the owner of this object.	true	true	true	false	false	false	invisible	nil	integer (4)	N/A
type	STI type	Type column that specifies the class of each row.	false	true	true	false	false	true	invisible	nil	string (255)	N/A
updated_at	Updated at	When this object was last updated.	true	true	true	false	false	false	visible	nil	datetime	N/A
updated_by	Updated by	The id of the last updater of this object.	true	true	true	false	false	false	invisible	nil	integer (4)	N/A
updated_by_user[login_name]	Updated by login	The login name of the User that updated this object.	true	true	true	false	false	false	visible	nil	string (255)	N/A
updated_by_user[person_name]	Updated by name	The name of the User that last updated this object.	false	true	true	false	false	false	visible	nil	string (255)	N/A
value	Value	The value which is stored for this API Data item	true	true	true	true	true	false	visible	nil	text (65535)	N/A

Related models:

Association name	Label	Description	Target Models	Type	Polymorphic?	Foreign key	Through	Min. Items	Max. Items
created_by_user	Created by	The creator of this object.	Global::UnrestrictedUser	belongs_to	false	created_by		0	1
owner	Owner	The owner of this object.	Global::UnrestrictedUser	belongs_to	false	owner_id		0	1
updated_by_user	Updated by	The last updater of this object.	Global::UnrestrictedUser	belongs_to	false	updated_by		0	1

Delete

- Required:
  - id
  - lock_version
  - _can_delete capability



GitHub, Inc. [US] https://github.com/workbooks/client_lib/blob/master/php/README.markdown#assertdelete-delete

assertDelete(), delete()

Delete one or more objects

Example:

```
$object_id_lock_versions = array(
  array (
    'id'                                  => $object_id_lock_versions[0]['id'],
    'lock_version'                        => $object_id_lock_versions[0]['lock_version'],
  )
);
$response = $workbooks->assertDelete('crm/organisations', $object_id_lock_versions);
// or: $response = $workbooks->delete('crm/organisations', $object_id_lock_versions);
```

Example: delete all order line items process button

```
/*
 * Clear out any old line items. Raise an exception on failure.
 */
function delete_all_line_items() {
  global $workbooks, $form_fields;

  $select_order_line_items = array(
    '_ff[]'                                   => 'document_header_id',
    '_ft[]'                                   => 'eq',
    '_fc[]'                                   => $form_fields['id'],
    '_select_columns[]'                       => array(
      'id',
      'lock_version',
    )
  );
  $workbooks->log('$select_order_line_items', $select_order_line_items);
  $response = $workbooks->assertGet('accounting/sales_order_line_items', $select_order_line_items);
  $delete_order_line_items = $response['data'];
  if (!empty($delete_order_line_items)) {
    $response = $workbooks->assertDelete('accounting/sales_order_line_items', $delete_order_line_items);
    $workbooks->log(count($delete_order_line_items) == 1 ? 'One order line item removed.' : count($delete_order_line_items) . " order line items removed.");
  }
  else {
    $workbooks->log('No order line items present.');
  }
} // delete_all_line_items()
```

Modifying data with the Wire Protocol

- You must supply an _authenticity_token (retrieved during login)
- For 'create' (for historic reasons) always specify a filter which selects no records ('id=0')
- Every array of parameters must be the same size
- Content-type:
  - Normally use application/x-www-form-urlencoded
  - Use multipart/form-data if you are uploading files

API Behaviour flags

- Pass these at login or per-request:
  - **_strict_attribute_checking** - set to true to reject requests to modify fields which do not exist (recommended)
  - **_time_zone** - override user-configured timezone

- Pass at login:
  - **json=pretty** (slows things down a little)
  - **_application_name**

Sending Email

- Sending email
  - e.g. send a report.
  - Uses the user’s email settings as configured in Workbooks to deliver.
  - API allows creation of drafts, or send immediately
  - API allows the use of email templates …. from email_send_example.php:

```
/*
 * Choose a template and a Case then use it to Send an email about the Case.
 */
$send_templated_email = array(
  'render_with_template_name' => 'Autotest Template',
  'render_with_resource_type' => 'Private::Crm::Case',
  'render_with_resource_id' => 2,
  'from_address' => 'from_address@workbooks.com',
  'to_addresses' => 'to.address1@workbooks.com, to.address2@workbooks.com',
  'cc_addresses' => 'cc.address1@workbooks.com, cc.address2@workbooks.com',
  'bcc_addresses' => 'bcc.address@workbooks.com',
  'status' => 'SEND',
);
$workbooks->assertCreate('email/emails', $send_templated_email);
```

Other Useful APIs

- Sending email
  - e.g. send a report.
  - Uses the user’s email settings as configured in Workbooks to deliver.
  - API allows creation of drafts, or send immediately
  - API allows the use of email templates

- Copy transaction document
  - Use the **create_from_id** parameter and specify the source document

- During login, specify **with_dropbox_email=true**
  - a dropbox will be created for the current user
  - bcc: it to send a copy into Workbooks and relate to items (e.g. by object_ref) automatically

- API Data
  - Useful to hold process ‘state’ between invocations.
  - Do work in small batches.

Synchronisation Hints and Tips

- Consider where it should run: on-premises or in Process Engine
- Typically run as a user with visibility of ALL records so ‘delta’ synchronisation can be reliable (examining and updating only recently-changed records)
    - Without visibility of everything, consider what happens if the set of visible records changes
- For efficiency, store external IDs in the ‘External Reference’ field (created_through_reference) rather than in a custom field. And set ‘Created Through’ field to your application name.
- Look for records where updated_at is on or after the latest updated_at previously considered
- Include the is_deleted field in your filter to see deleted records alongside undeleted records
- Make sure you trim() fields to fit
- Consider conflicting edits on each system
- Don’t stall the entire sync just because a single record cannot be synced
- Sync in batches, saving state/progress frequently
- Log object_ref whenever possible
- Email addresses do not have to be unique in Workbooks, they do in some other systems

workbooks_api.php

github.com/workbooks, choose client_lib/php

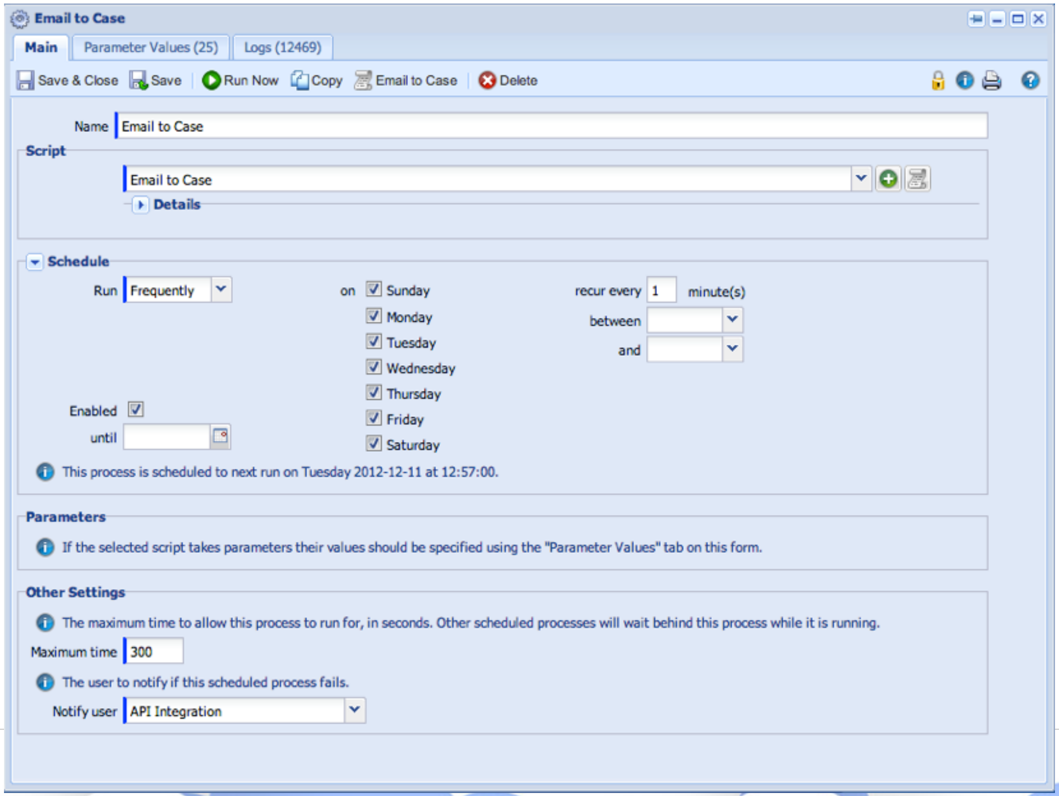
- This is a worthwhile piece of code to review to understand how it uses the wire protocol.
  - Key functions:
    - assertCreate(), assertGet(), …
    - which call: create(), get(), …
    - which call: api_call()
    - which calls: make_request()

Files

- If a field is a file then use multipart/form-data content type for the transfer.
- Create attachments using the ‘resource_upload_files’ endpoint.
- Often fetch back using ‘upload_files’ endpoint.
- See the ‘Case Portal’ (in the script library) for an example in PHP. Or upload_file_example.php
- workbooks_api.php will give you a hint.
- Attaching files to email not currently possible through the API. Instead you must build your email the hard way using ‘rfc822’ format.

Scheduled Process

- Restriction: only one per database can run at a time
  - Duration should be small
- Exit Code matters
  - 0 => OK
  - 1 => Retry later
  - 2 => Failure
- Upon failure:
  - Process disabled
  - User notified



Name: Email to Case

Script: Email to Case

Details

Schedule

Run: Frequently

on: ☑ Sunday ☑ Monday ☑ Tuesday ☑ Wednesday ☑ Thursday ☑ Friday ☑ Saturday

recur every 1 minute(s)

between

and

Enabled ☑

until

This process is scheduled to next run on Tuesday 2012-12-11 at 12:57:00.

Parameters

If the selected script takes parameters their values should be specified using the "Parameter Values" tab on this form.

Other Settings

The maximum time to allow this process to run for, in seconds. Other scheduled processes will wait behind this process while it is running.

Maximum time: 300

The user to notify if this scheduled process fails.

Notify user: API Integration

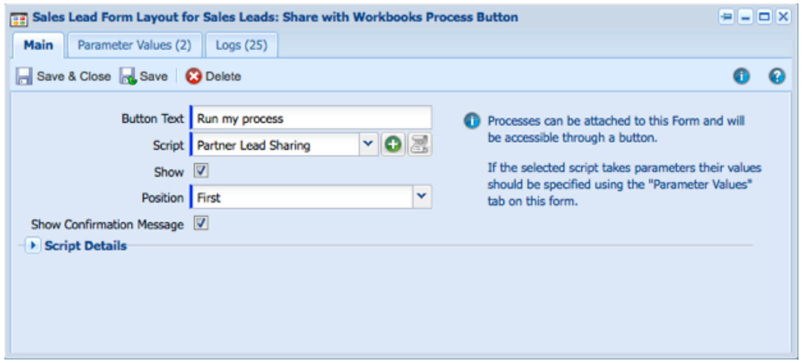
Exercise: Case Follow-up email

- Create a custom field which records if a follow-up email was sent
- Find all cases in 'Awaiting customer response' state which have not had a follow-up email and send a reminder to the primary contact of each.
- As each email is sent, record that the email was sent

Process Button

Workbooks.com

- Added via Custom Form Layout.
  - Automation tab
- Buttons appear on record toolbar.
- Process invoked after successful validation and save of the record.
- Process completes before form reloads.
- Form fields passed to the process in $form_fields array.
- Summary shown as an alert message unless turned off.
- Button processes can be attached to the record save action to run every time.



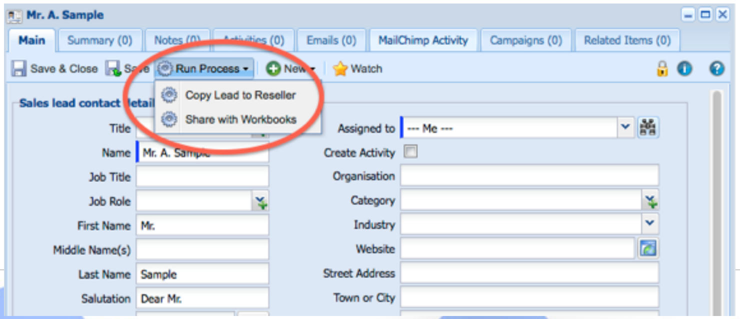
Sales Lead Form Layout for Sales Leads: Share with Workbooks Process Button

Button Text	Run my process
Script	Partner Lead Sharing
Show	☑
Position	First
Show Confirmation Message	☑

Script Details

Processes can be attached to this Form and will be accessible through a button.

If the selected script takes parameters their values should be specified using the "Parameter Values" tab on this form.



Mr. A. Sample

Sales lead contact details

Title		Assigned to	--- Me ---
Name	Mr. A. Sample	Create Activity	☐
Job Title		Organisation	
Job Role		Category	
First Name	Mr.	Industry	
Middle Name(s)		Website	
Last Name	Sample	Street Address	
Salutation	Dear Mr.	Town or City	

Other Process Types

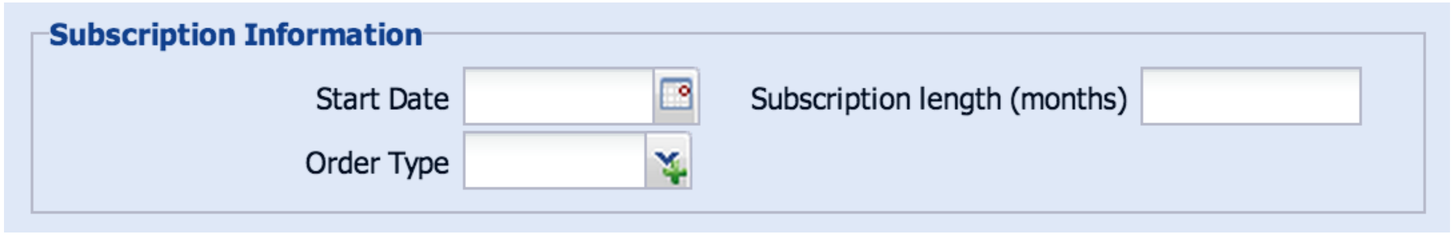
- Recent releases of Workbooks have added various options
  - You can attach a process button to a report.
    - Runs a process on a list of items (like bulk update).
- Others will come soon

Exercise: Process Button Case Reminder

- Take the code you wrote to send a reminder about a Case and make it run for the current Case using a Process Button.

Exercise: Order Configurator

- Create the following custom fields in a section for Orders:



Subscription Information

Start Date		Subscription length (months)	
Order Type			

- Create a simple Order Configurator:
  - A process button should interpret those fields and create a set of line items.
  - e.g. for a 6-month subscription create 6 line items, at monthly intervals.

Workbooks Data Model

- Main Objects
  - People/Organisations/Parties
  - Leads
  - Opportunities/Quotations/Orders/Invoices/Credit Notes
    - Line Items
  - Marketing Campaigns, Members, Statuses
  - Cases
  - API Data
- Custom Fields
- Queues
- Relationships
- Users, Groups, ACLs
- Search

Some Examples

- Order Configurator
- HubSpot Synchronisation
- Case Portal - portals in general
- Box Integration
- MailChimp Synchronisation - checkpointing
- Email to Case

Workbooks.com

API Gotchas

- **Don't…**
    - Cache cookie values between multiple requests
        - Always send back the value of the cookie just received
    - Do your own JSON parsing!
    - Assume ordering of response fields - they are unordered
    - Implement your own HTTP stack. Use the one from your framework
    - Forget to escape HTML entities
    - Forget to consider locale: character sets, timezone
    - Leave performance as an afterthought; test with realistic data volumes

Ensuring UI responsiveness

- Do not do very much at all in process buttons
  - Do the heavy lifting in scheduled processes - consider a implementing a queue of work in API Data
- Checkpoint frequently to break up long-running processes

In general

- Filters: filter on indexed fields, ideally on the main table and use ‘eq’ or ‘starts with’
- Do not run repetitive processes any more frequently than necessary
- Don’t update and refetch the same record multiple times: collect your changes together and apply in a small number of API calls

Error handling

- If you are using a binding, use the assert…() method
- What happens if your script fails halfway through?
  - Catch exceptions where necessary and clean-up to leave a consistent state and/or log
- Expect errors
  - Timeouts and system errors will happen from time to time
- Stale object errors
  - Consider retrying the update or delete
- Consider using per_object_transactions
  - On failure, the whole request will roll back

Support

- The API changes from time to time
  - Features are added, e.g. the proportion of Workbooks which is accessible via the API increases.
  - All changes are backwards-compatible.
    - Any exceptions would be widely announced before reaching.
  - All published examples are auto tested
- Look at examples - on github.com/workbooks and in the Script Library
- Contact us via support@workbooks.com
  - Please include your code, the intention of the script, and as much information about the problem.
  - Make sure you've read your logs carefully first. Make sure you log well.
  - We really like tidy code which is easy to read and well formatted.
  - We are happy to write scripts for our customers if you purchase Admin Credits from us: contact sales@workbooks.com