Intersect Australia Ltd
PO Box H58, Australia Square
Sydney NSW 1215

T +61 2 8079 2590

ABN 67 131 752 657

DIVER Release Notes

Release version 2.3.02

Cathy Chamley | 4 December 2015 | 0.1

Contents

Intersect Australia Ltd
PO Box H58, Australia Square
Sydney NSW 1215

T +61 2 8079 2590

ABN 67 131 752 657

1 Overview

The DIVER application is a Ruby on Rails application initially built by Intersect Australia for the Hawkesbury Institute for the Environment (HIE) as part of an an ANDS-Funded Data Capture Project - DIVER - Climate Change and Energy Research Facilities.

The resultant HIEv data capture system has been in production at UWS HIE since February 2013. The application is also in production at Macquarie University, Faculty of Business and Economics to support the Faculty's marketing research and other research projects.

Intersect is developing DIVER as a general-purpose data capture product offering know as DIVER. For more information, see http://www.intersect.org.au/content/intersect-diver.

The current release version is 2.0.01. This version, 2.3.02, has been funded by Western Sydney University to support HIE's growing data sharing and collaboration needs.

Introductory information about DIVER can be found here: https://github.com/IntersectAustralia/dc21-doc/blob/master/README.md.

Source code is available at: https://github.com/IntersectAustralia/dc21.

2 System Requirements

The recommended minimum environment for hosting the web application is as follows.

- A Linux server, preferably RHEL6.x/CentOS 6.x (el6) with the standard "Server" packages installed.
- 4 cores, contemporary chipset +2.5GHz.
- 4GB of RAM.
- Postgresql (8.1.x or later)

NOTE: Additional cores and RAM may be required if multiple concurrent users download large files simultaneously.

3 Installation / Upgrade Instructions

Server-side

- If you are installing DIVER for the first time, follow these instructions: https://github.com/IntersectAustralia/dc21-doc/blob/2.1.02/Deployment_Guide_-_First_Time_Server_Build.md
- If you are upgrading from version 2.0 to Version 2.1, follow these instructions:

  "Updating Your System" in the page https://github.com/IntersectAustralia/dc21-doc/blob/2.1.02/Deployment_Guide_-_Deploying_A_New_Version.md
- If you are upgrading from version 1.9 to version 2.1, the upgrade needs to be done in two steps:

  1. Follow these instructions: https://github.com/IntersectAustralia/dc21-doc/blob/2.1.02/Upgrading_From_1.9.04_to_2.0.01.md
  2. Then follow the instructions above to upgrade from version 2.0 to version 2.1.

NOTE: The tag for this release is as per the Release version above. This tag must be selected during the installation process (the install process will request that you select a tag from a list of available tags). If later tags are available, choose the latest "2.1.nn" tag. If in doubt, you should check with the Intersect development team (see http://www.intersect.org.au/contact) as to which tag should be used.

NOTE: If you are upgrading, you should not attempt to run client-side API scripts until the client-side automated upload scripts have been upgraded too. For details, see Client-side Automated Upload section below.

Client-side Automated Upload

If you are installing the DIVER client-side API for the first time, refer to: https://github.com/IntersectAustralia/dc21-doc/blob/2.1.02/Setting_Up_Automated_Load_From_PC.md.

Alternatively, if you are upgrading from version 1.9 or version 2.0, you don't need to re-run the setup scripts. Instead, you should follow these instructions:

1. Download the API zip file (see above link to do the download)
2. Extract the IntersectAustralia-restful-api-uploader-xxxxxxx folder and place this folder in the same folder as the existing IntersectAustralia-restful-api-uploader folder. (xxxxxxx is a unique 7 digit hex string).
3. Use Notepad to edit the windows_api_load.bat file "ruby" command to refer to the new IntersectAustralia-restful-api-uploader-xxxxxxx folder. For example:

   ruby "C:\scripts\DIVER\IntersectAustralia-restful-api-uploader-2c20fa8\lib\run_wrapper.rb" ....
4. You should now be ready to run the client-side API

NOTE: There are new optional configuration parameters available for the transfer_config.yml file. Refer to the above link, "Edit config file" section, see new access control parameters. Access control is fully described in the Version 2.3 User Manual.

4 Required Changes to Client Software that uses the API

All changes made to the API in v2.3 are backwards compatible for any expected usage. This release includes a small bug fix made so that unauthorized updating of TOA5 files is prohibited – but this will not affect any users who are using the API in accordance with the intended usage.

5 Known Issues and Limitations

- Frequently adding large numbers of files (>1,000) and data (>1GB) to the cart and invoking the Download or Package functions is not recommended as these actions may impact on system performance.
- The cart list is not paginated and may become unusable if more than 1,000 files are added to it.
- The API search function returns an un-paginated json list. This list may become very large (and may take some time to download) if API search is unfiltered.
- Manual file downloads fail when run with Internet Explorer Version 11 on Windows Surface tablet. Internet Explorer Version 11 is not currently supported.
- The automated API upload has a time-out set to 30 minutes for individual file uploads. This means that uploads of large files over a slow network thus taking longer than 30 minutes will not complete successfully. There is a workaround: The time-out can be increased by editing the first line of the file_upload.rb file in the lib folder of the installation download.

6 Test Report

Intersect Test Environment

Full QA and testing was completed for all stories completed by 22.09.15. Details of testing is included in individual Jira items, and auto-tests were added for all items, or Manual regression tests were updated for items which couldn't be auto-tested.

A full Manual Regression test was run on 22-9-15 to assure the new and existing functionality up to that date. Auto-tests were conducted again on the completion of Post UAT Development to assure all development and previous functionality. While story and unit testing were performed for Post UAT Development, a full manual regression test was not performed at that time as the changes were minor and contained.

Intersect QA/Testing approach

1. Specifications are written with testing in mind.
2. Developers do unstructured unit tests, then when ready they release to QA environment where testers can verify changes.
3. QA/Testers test against the Jira story – This includes structured & unstructured testing, and any feedback is discussed with Developers/ P.O.'s as required. New Jiras are raised as required for found issues.
4. QA/Testers upgrade Manual regression tests.

5. Automated tests are run through Jenkins to assure System integration testing Code freeze.
6. Final release End to end Manual Regression testing is conducted to assure full functionality.
7. Testing of the User Manual.

QA/Testing Results

Component	Number of tests Executed	Result (Pass %)	Comments
Admin & General Functionality & Usability	10	100%	Raised a minor bug relating to UI. https://jira.intersect.org.au/browse/UWSHIEVMOD-110
File Management	3	100%	
Uploads, Cart, Packaging, Downloads & Publishing	6/7	100%	Re-wrote long test, currently incomplete windows automatic upload script test in progress.
Search Test	1	100%	
Rescue Workers	2	100%	
Package API	1	0%	Raised a bug that API seems not to warn if a badly formatted date provided: https://jira.intersect.org.au/browse/UWSHIEVMOD-112