
APPortal Documentation

Release 0.0.1

Geoff Williams

Jun 29, 2020

Contents

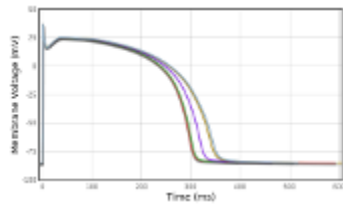
1	IMPORTANT	3
2	Preamble	5
3	Installation	7
3.1	Installation	7
4	Security	59
4.1	Security	59
5	Personal Data	63
5.1	Personal Data	63
6	Configuration	65
6.1	Configuration	65
7	Maintenance	67
7.1	Maintenance	67
8	General	83
8.1	General	83
9	Tools	85
9.1	Tools	85
10	FAQ (Frequently Asked Question)	91
10.1	FAQs	91
11	TroubleShooting	93
11.1	Troubleshooting	93
12	Glossary	97
12.1	Glossary	97
	Index	99

Project Home https://bitbucket.org/gef_work/ap_predict_online

Documentation <http://apportal.readthedocs.io/>

Created Jun 29, 2020

Version 0.0.1



CHAPTER 1

IMPORTANT

As of early 2019 this application, originally created at the University of Oxford's Department of Computer Science, is no longer being actively developed.

Subsequent activity has been transferred to the development of a newer version, AP-Nimbus at <https://github.com/CardiacModelling/ap-nimbus> by the University of Nottingham's School of Mathematical Sciences .

CHAPTER 2

Preamble

- User/Scientific documentation is accessible via <https://chaste.cs.ox.ac.uk/ActionPotential/about>.
- In many places the documentation may appear generalised as some aspects of the AP-Portal's operational dependencies, e.g. databases, Java servlet containers, etc., are *predominantly* unconstrained. Anyone intending to use the portal should be familiar with installing and operating the dependencies independently.
- There is no documentation herein on how to install ApPredict, which is the portal's underlying cardiac simulator. For such instructions please visit <https://chaste.cs.ox.ac.uk/trac/wiki/ApPredict>. Alternatively, there are some sample installations beneath the AP-Portal repository [install](#) directory.
- Full source code, Javadocs, and limited additional documentation is available at https://bitbucket.org/gef_work/ap_predict_online. It is anticipated that this documentation will be the principle source of technical information, which can be accessed at <http://apportal.readthedocs.io/> and via that resource, exported to a number of formats.

3.1 Installation

3.1.1 Deployment options

Decision chart

Note: In all cases a ‘Code User’ or ‘Code Developer’ version” of ApPredict also needs to be installed (see [official ApPredict build instructions](#) and [sample ApPredict install scripts](#))!

Note: The names `app-manager`, `business-manager`, `business-manager-api`, `client`, `client-direct`, `dose-response-jni`, `dose-response-manager` and `site-business` are all references to the *generic portal components* of AP-Portal (`client` and `client-direct` both require `client-parent` and `client-shared` to be installed).

```

+-----+
| Do you intend to |
| manually enter   |
| IC50/pIC50 values? |
+-----+
|                 |
| Yes             | No
|                 |
+-----+ +-----+
| Install app-manager | | Do you intend to |
| and client-direct only | | install the demo |
+-----+ | system only? |
|                 |
|                 |

```

(continues on next page)

(continued from previous page)

Yes	No
+-----+	+-----+
Install app-manager,	Do you have dose-response data
business-manager,	available to use?
business-manager-api,	+-----+
client, dose-response-jni	
dose-response-manager	No
and site-business	Yes
+-----+	+-----+
	Install app-manager,
+-----+	business-manager,
Install app-manager,	business-manager-api and
business-manager,	client.
business-manager-api, client,	Create your own site-business
dose-response-jni and	+-----+
dose-response-manager.	
Create your own site-business	
+-----+	

Options diagram

The illustration below depicts typical installation options (*Option A* and *Option B*) of all the generic AP-Portal components in two scenarios :

Option A

The simplest installation consisting solely of the `client-direct` and `app-manager` components.

This configuration expects users to manually enter compound IC50, Hill and saturation values in the portal interface. Simulation results are stored in `client-direct`.

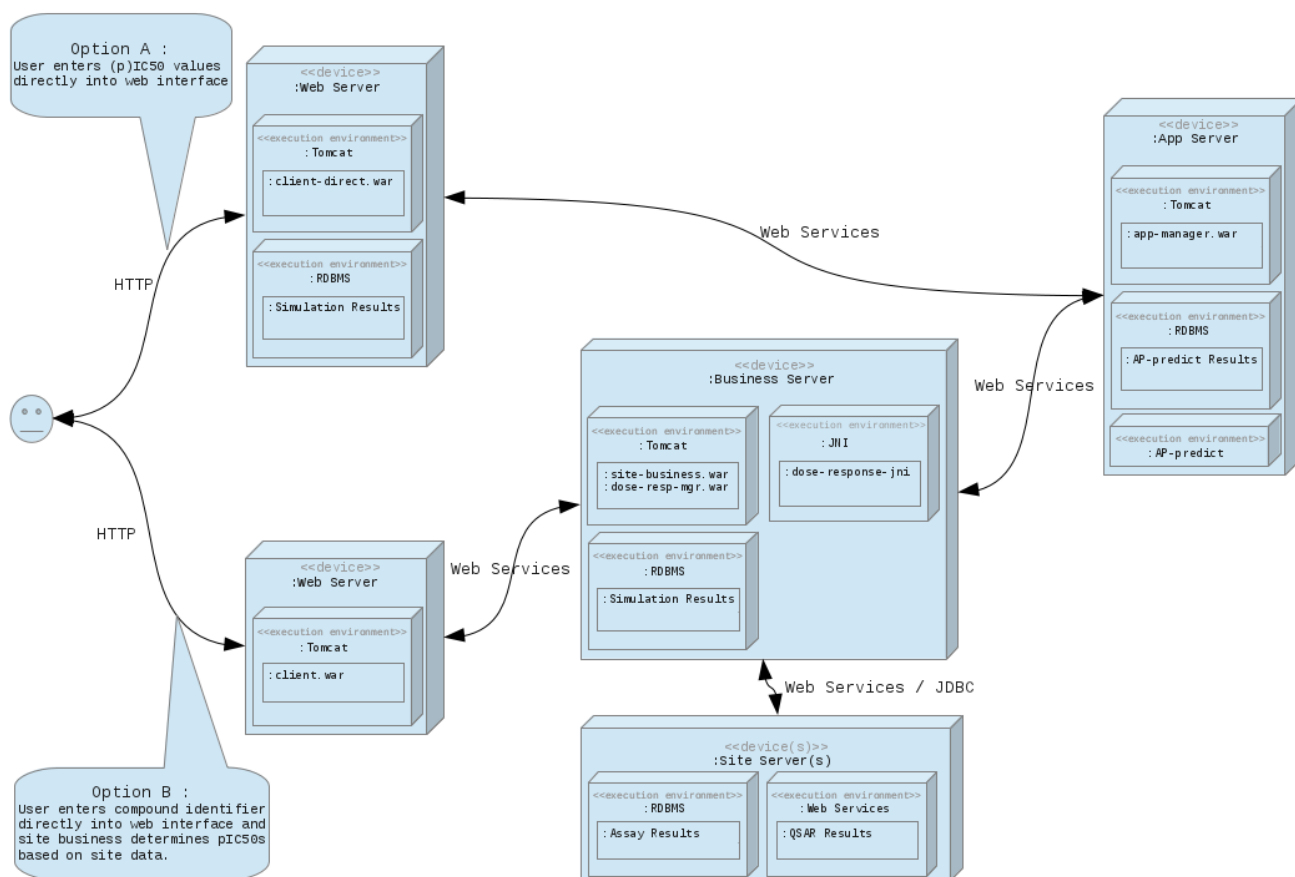
Option B

The more intricate installation involves the interaction of `client`, a `site-business`, `dose-response-manager` (optional) and `app-manager` components all working together to query *local* compound databases to retrieve compound IC50, Hill and other values across a range of assays.

Warning: A `site-business` must be installed. If you wish to see your own compound data then you will need to *create a bespoke* `site-business`.

3.1.2 Minimum requirements

In order to build and host the AP-Portal **AND** the required ApPredict application the following are the general minimum requirements.



Logo goes here.

Notes ⓘ

Please do not enter commercially sensitive information in this field!

Cell Model ⓘ (Please contact us via the [Contact](#) page if you would like a different model)

Model

TenTusscher ⌵

 TenTusscher et al. (2006) human ventricular cell model (epicardial)

View all Model Details

Pacing Details

Pacing frequency

1.0

 (Hz) ⓘ Frequency of pacing

Maximum pacing time

5

 (mins) ⓘ Maximum pacing time

Ion Channel Current Inhibitory Concentrations

Note 1 : No value for Inhibitory Concentration implies "no effect"

Note 2 : Unless otherwise assigned Hill Coefficients default to 1, and Saturation Levels default to 0.

Ion current	IC50 ⌵	M ⌵ ⓘ	More options +	Channel protein	Gene	Description
IKr	<div></div>	M		K _v 11.1	<i>hERG</i> or <i>KCNH2</i>	Rapid delayed rectifier potassium current
INa	<div></div>	M		Na _v 1.5	<i>SCN5A</i>	Fast sodium current
ICaL	<div></div>	M		Ca _v 1.2	<i>CACNA1C</i>	Long-lasting (L-type) calcium current
IKs	<div></div>	M		K _v 7.1	<i>KCNQ1/minK</i>	Slow delayed rectifier potassium current
IK1	<div></div>	M		K _j 2.1	<i>KCNJ2</i>	Inward rectifier potassium current
Ito	<div></div>	M		K _v 4.3	<i>KCND3</i>	(Fast) transient outward potassium current

Compound Concentrations

Min.

0

 (μM) ⓘ Minimum compound concentration

Max.

10000

 (μM) ⓘ Maximum compound concentration

Intermediate point count

4

 ⓘ Count of plasma concentrations between the minimum and maximum

Intermediate point log scale ☒ Whether intermediate points should be using a log scale

Run Simulation

Reset

Logo goes here.

[Home](#) | [About](#) | [Contact](#) | [Log Out](#)

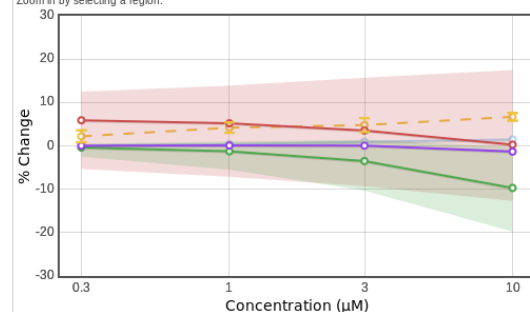
Cardiovascular AP-portal Report for cmpd1

Compound Identifier:

Overall Progress ☐ QSAR@0.5 (Hz) ☒ Quattro,FLIPR,Barracuda@0.5 (Hz) ☒ PatchXpress,Qpatch@0.5 (Hz) ☒ ManualPatch@0.5 (Hz) ☒

CellML Model : Shannon (Shannon et al. (2004) rabbit ventricular cell model).

Zoom in by selecting a region.



Source

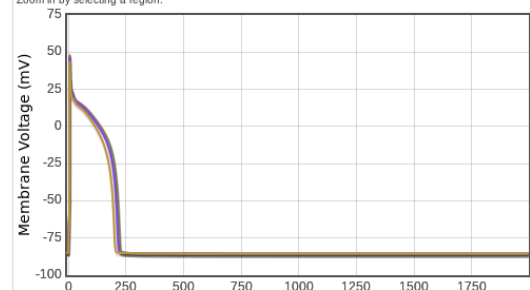
- ☒ Experimental QT @ 0.5 Hz
 - ☒ QSAR@0.5 (Hz)
 - ☒ Quattro,FLIPR,Barracuda@0.5 (Hz)
 - ☒ PatchXpress,Qpatch@0.5 (Hz)
 - ☒ ManualPatch@0.5 (Hz)
-

AP-predict input data [All values IC50 (μM) of the mean pIC50 of any modifier]:

Assay (or Assay Group)	hERG	CaV1_2	NaV1_5	KCNQ1
QSAR	25.119 (QSAR)	79.433 (QSAR)	63.096 (QSAR)	Not available
Quattro,FLIPR,Barracuda	50.119 (Quattro_FLIPR)	243.447 (Quattro_FLIPR)	87.322 (Quattro_FLIPR)	50.119 (Quattro_FLIPR)
PatchXpress,Qpatch	19.953 (Qpatch)	25.119 (PatchXpress)	87.322 (Quattro_FLIPR)	50.119 (Quattro_FLIPR)
ManualPatch	22.387 (Manual Patch)	50.119 (Manual Patch)	39.811 (Manual Patch)	50.119 (Quattro_FLIPR)

Conc. μM
Change %

Zoom in by selecting a region.



Simulation and concentration (μM)

- ☒ 0 [μM]
- ☒ PatchXpress,Qpatch@0.5 (Hz) @ 0.001 [μM]
- ☒ PatchXpress,Qpatch@0.5 (Hz) @ 0.3 [μM]
- ☒ PatchXpress,Qpatch@0.5 (Hz) @ 1.0 [μM]
- ☒ PatchXpress,Qpatch@0.5 (Hz) @ 3.0 [μM]
- ☒ PatchXpress,Qpatch@0.5 (Hz) @ 10.0 [μM]

Hardware

Servers

Note: Minimum 1 (!?)

Although 2 would be useful to isolate the operations of the `app-manager`'s compute-intensive `ApPredict` simulation invocations if many concurrent simulations are anticipated.

All inter-component communication (e.g. `client-direct` <--> `app-manager`, or `site-business` <--> `app-manager`) is via WS (Web Service) communication so components do not need to reside on the same hardware.

Internet access

Without direct internet access there will be a significant amount of copying of files from one machine to another during `AP-Portal` and `ApPredict` installation.

Direct internet access preferred for :

1. Downloading `AP-Portal` source code from [bitbucket](#).
2. Downloading `ApPredict` source code from [github](#).
3. Downloading library dependencies for `AP-Portal` and `ApPredict`. For example, during Maven building of `AP-Portal`, and for Python package and CHASTE [requirements](#) for building of `ApPredict`.

Once built and installed there is no further need for direct internet access except by `ApPredict` for downloading *variability lookup tables* (see also *Variability Configuration*).

Processing cores

Note: Minimum 4, preferably 8-16.

In theory just 1 but that would be impractical as it would take considerable time to install the software and operationally the application, once in use, would become frequently unresponsive.

A single `ApPredict` simulation can take usually between 3 to 7+ minutes, depending on aspects such as :

1. The ion channel model (i.e. some run quicker than others),
2. The IC50 values and how quickly a “steady state” is achieved.

If `client-direct` is being used then there is only a single invocation of `ApPredict` for each simulation run. If however `client` is being used then there will be an invocation of `ApPredict` for each combination of :

1. Frequency, e.g. 0.5Hz, 1Hz
2. Assay (or Assay Group), e.g. Barracuda, PatchXpress/Qpatch.

Potentially therefore `app-manager` could be invoking 10+ concurrent `ApPredict` simulations if there are more than one frequency and multiple assays – each simulation potentially consuming 100% of a core.

The actual number of maximum concurrent simulations is configurable in `app-manager` such that some cores can remain free to service undemanding `client` or `client-direct` requests if hosting everything on one machine.

Hard disk space

Note: Minimum 50 Gb, preferably 500Gb+¹.

Item(s)	Approx. min. size
CHASTE dependency libs, bins, includes, etc.	800Mb
CHASTE + ApPredict source code	3Gb
ApPredict <i>variability lookup tables</i>	5Gb ²
AP-Portal PK processing capability	Up to 100Mb per PK sim!

Non-PK and non-*variability* simulations generally consume very little hard disk space or RAM (Random Access Memory), however since late 2017 the ability to run both PK and *variability* simulations in the `client-direct` component has significantly increased the operational memory requirements of AP-Portal operations. There is also now available a much-expanded range of *variability lookup tables* (see *Variability Configuration*).

RAM

Note: Minimum 8Gb, preferably 12-16Gb+.

A large amount of RAM is required during the build operations as part of the installation of AP-Portal and ApPredict, particularly if building using multiple cores.

During runtime RAM can also be consumed during simulation invocation if *variability* is being calculated, as each simulation may attempt to load a 700Mb+ file into RAM.³

Software

Linux (or similar)

The majority of AP-Portal components, being Java-based, **could** (i.e. it hasn't been tested) run on Windows, but not the `app-manager` component or ApPredict.

Java 7+ JDK (Java Development Kit)

For example, [Oracle JDK](#), [OpenJDK](#) or similar.

Warning: Since early 2019 `client` and `client-direct` require Java 8 to run.

¹ With the arrival of PK (Pharmacokinetic) processing (early 2018) there has been a commensurate increase in the hard disk space required as there is now the ability to upload PK data files in the region of 40Mb - these files, and the large results datasets they generate, are persisted in the database and so over time the amount of memory consumed may rise considerably if such PK simulations are not manually deleted after the results have been downloaded.

² Originally there were only a couple of *variability lookup tables* (which can be 700Mb+ in size) available for ApPredict - however since late 2017 there are a number of these files available which should ideally be available on the local filesystem (see also *Variability Configuration*).

³ Originally *variability* processing had been available only in the `client` component but since late 2017 it has been possible to also do it in the `client-direct` component.

Warning: Java PermGen errors sometimes occur during *Maven* building when using Java 7 - so whilst not essential, perhaps it would be better to use Java 8.

Database

Warning: AP-Portal has been running on *MySQL*, *Oracle* (10g) and *HSQL* databases. Other databases would be possible but their configuration would require the modification of files which are currently under version control, i.e. they have not yet been made locally modifiable.

Used to persist data, e.g. *MySQL* or similar.

Apache Maven

The AP-Portal component build mechanism.

Servlet container

Used to host the AP-Portal components, e.g. *Tomcat* or similar.

Git

Version control system used by AP-Portal and ApPredict.

ApPredict

Sample build operations, which allow for versioned building of ApPredict (and Chaste), are available for *CentOS 6.5* and for *Fedora 20*.

Warning: Building ApPredict (or more specifically the collection of dependencies - invariably referred to as “*chaste-libs*”) is potentially a complex and time-consuming task. Once the dependencies have been built it can be relatively straightforward to (re)build ApPredict.

Skills

Linux command line

Some aspects of AP-Portal building and installation can be assisted by using a tool such as the Eclipse IDE (Integrated Development Environment), but some aspects of app-manager installation (which handles ApPredict invocation) in particular do require a good knowledge of command line operations.

Building ApPredict requires a good understanding of command line operations.

Java and Spring

AP-Portal is a Java-based application which makes extensive use of the [Spring Framework](#) and its various [projects](#).

If you wish to create your own `site-business` then it is inevitable that you will need to be proficient in Java and potentially [Spring](#).

XML (eXtensible Markup Language)

Some of the AP-Portal configuration files are XML-based and some knowledge of XML document structure and complimentary technologies would probably be appropriate.

If you wish to create your own `site-business` then it is inevitable that you will need to be comfortable with XML.

Databases and SQL (Structured Query Language)

Whilst various components require interactions with databases – requiring an understanding of their installation and operations – an in-depth knowledge of SQL is not necessary. Generally the more complex SQL used by AP-Portal is derived from using [Hibernate](#), although on occasions it may be appropriate to understand some of the database configuration options.

If you wish to create your own `site-business` then it is very likely that if they are SQL-based then a corresponding level of SQL understanding would be required for bespoke database interaction to your compound databases.

Servlet Containers

AP-Portal is dependent on servlet container software, e.g. [Tomcat](#), and therefore familiarity with the configuration and operation of these is required.

System user privilege level

No component of AP-Portal requires elevated (e.g. `root`) privileges for either installation or operation, although this does restrict the port numbers which `client` and `client-direct` components can listen on some systems and may make for unsightly URL (Uniform Resource Locator)'s.

Similarly, if the application is installed as a non-privileged user then on some systems it may not be straightforward to have the components and dependent applications automatically started if the native system is rebooted. For example, `cron's @reboot` may not work on some systems. In such cases it may require the assistance of someone with elevated privileges to create, for example, a `SysV init` script.

Guidance Checklist

'R' - Required; 'D' - Desired; 'O' - Optional

Base checklist

	Requirement	
R	<i>Hardware</i>	
R	<i>Software</i>	
R	<i>Skills</i>	
R	User account ¹ on host server	
O	Component restart on reboot, e.g. <code>cron's @reboot, SysV</code>	

For a bespoke site-business installation

See also:

bespoke site-business installation

	Requirement	
R	Read-only site assay database account for AP-Portal	
R	Determine assays (e.g. PatchXpress, QSAR (Quantitative structure–activity relationship)), assay grouping	
R	Determine if <i>dose-response data</i> fitting required	
R	Determine ion channels, e.g. CaV1.2, hERG	
R	Determine CellML models (and the default) to use, e.g. Shannon, ten Tusscher	
R	Determine default units, e.g. pIC50, IC50	
R	Determine strategies for selecting pIC50 ² to use	
R	SQL ³ to retrieve assay, ion channel, etc., data	
R	Details of services, e.g. WSs, which supply additional data	
R	IT (Information Technology) personnel to implement Java/ <i>Spring</i> code	

For a bespoke client-direct installation

See also:

bespoke client authentication

	Requirement	
D	Company logo image ⁷	
D	Company login and logout mechanism ⁷	
D	Company “contact” information (e.g. <i>local</i> contact emails) ⁷	
D	X.509 certificates (optional self-signed ⁴) for HTTPS (Hypertext Transfer Protocol (Secure))	
O	Generate and populate usernames and passwords SQL file ^{5,7}	

¹ No need for elevated privileges user account on the native filesystem (although need to consider auto-restarting component services or use of TCP ports 80 and 443!).

² Different sites have data of varying characteristics requiring specific actions.

³ Assuming that the data is held in an SQL database!

⁷ These operations should take place in `client-shared`.

⁴ Use of self-signed certificates generate threatening browser warnings!

⁵ Not “Required” if you’re just trying things out!

For a bespoke client installation

See also:

bespoke client authentication

	Requirement	
D	Company logo image ⁷	
D	Company login and logout mechanism ⁷	
D	Company “contact” information (e.g. <i>local</i> contact emails) ⁷	
D	X.509 certificates (optional self-signed ⁴) for HTTPS	
O	IT personnel to implement Java/ <i>Spring</i> code ⁶	
O	Generate and populate usernames and passwords SQL file ⁵⁷	

3.1.3 AP-Portal extensibility

Before considering installation options for the *generic portal components* there are a some things which may require consideration if you intend to have a *site-specific* installation of the portal. This ambition will require additional on-site work which is likely to require a lot of organisation, development and support time to create a bespoke implementation.

Below are the main areas of *site-specific* preparation:

User authentication and authorisation

Brief outline

How users are authenticated by the `client` and `client-direct` components takes one of two techniques provided by their parent `client-shared` component. At runtime the technique used is determined by the use of *Spring profiles*.

1. “prepopulated user authentication”

The *off-the-shelf* portal authentication mechanism which makes user of manually populated user databases.

This mechanism is relatively straightforward, in that the `users.sql` files (see *client* and *client-direct*) containing usernames, passwords and *roles*, are created and populated during portal installation and manually updated thereafter.

Only when using this option do potential portal users have the ability to *register* to use the portal using the portal interface.

2. “bespoke user authentication”

The provision of a bespoke authentication mechanism (e.g. NTLM (NT LAN Manager), *Active Directory*) by the site itself.

Whilst the authentication mechanism may already be available on-site, there will need to be a bespoke *filter* (see *Request Filtering*) created which interfaces with this mechanism.

⁶ If you wish to restrict access AP-Portal

Task responsibility (for “*bespoke user authentication*”)

The primary objective of this activity most likely involves the IT personnel determining how to control access to the `client` and `client-direct` user interfaces by intercepting requests (see [Request Filtering](#)) to determine user identity and return a browser redirect to site authentication mechanisms before the user is granted access.

The IT people involved will need to have some experience of Java software development (including [Spring](#)), as well as XML formats.

Additional detail

Request Filtering

The `client` and `client-direct` components both make use of [Spring](#) Security’s `springSecurityFilterChain` `FilterChainProxy` by virtue of the `src/main/webapp/WEB-INF/web.xml` filter naming. This mechanism intercepts all incoming requests and passes control to a variety of filters which can take whatever action is appropriate for the circumstances, e.g. accept (and pass the request to the next filter in the chain), or reject (and perhaps show a default page or redirect to an authentication mechanism).

Amongst these filters there are some standard checks such as for security (e.g. CSRF (Cross-Site Request Forgery)) and valid sessions, but there can also be *site-specific* checks which can, for example, check for the presence of site cookies or query *local* authentication/authorisation mechanisms, e.g. NTLM. Equally, some portal pages could be deemed to be available for perusal by any visitor, e.g. such as in `appCtx.noSecurity.xml`

If a request arrives at a portal from an unauthenticated user hoping to visit restricted access areas then a *site-specific* filter (e.g. `sample.appCtx.sitePreauthFilter.xml`) injected into the `springSecurityFilterChain` (e.g. `custom-filter` in `sample.appCtx.bespoke.xml`) will reject the request and most likely forward the user to the site’s authentication mechanism.

Authorisation mechanisms

Access to certain areas of the portal are controlled by the “*Role*”s which were assigned to the user at authentication time.

As a general rule all users accessing the `client` or `client-direct` portals must, unless they are accessing unrestricted areas, be in the `ROLE_USER` group (e.g. as determined by `sample.appCtx.bespoke.xml` or `appCtx.prepopulated.xml`).

Currently the only supported roles are defined in `Role.java`

- `ROLE_USER` Generally every portal visitor must be authenticated and assigned this “*Role*”.
- `ROLE_POWER_USER` Generally assigned to users with access to certain “elevated privileges” functionality, such as entering ion channel spread values, or uploading CellML files.
- `ROLE_ADMIN` Assigned to the person(s) responsible for administrative portal duties.

See also:

[Portal feature access](#)

User registration

Brief outline

If you are keen to allow users to register themselves to use AP-Portal in either of the `client` or `client-direct` interfaces then there is the option to interact with *local* SMTP (Simple Mail Transfer Protocol) facilities and use Google's *reCAPTCHA* facilities to improve the likelihood that the registrations are from genuine potential users.

Note: The option to have a registration mechanism only applies when the “*prepopulated user authentication*” *user authentication mechanism* is active. If a “*bespoke user authentication*” mechanism is used then there is no option to display a registration form in either of the `client` or `client-direct` interfaces.

Note: “Registration” in AP-Portal represents a portal administrator being notified, via email, of a user's desire to use the system. It does not provide features such as dynamic user database population, user password selection or “remind me” features common on most web sites.

Task responsibility

The primary objective of this activity most likely involves the IT personnel determining how to configure interaction with the *local* SMTP server and, if using *reCAPTCHA*, how to request the necessary keys.

Additional detail

Scenarios related to registration :

- *Local* SMTP service configured.

Visitors to the `client` and/or `client-direct` interfaces who wish to register can complete a registration form (which prompts for a contact email address) which when submitted uses the configured SMTP service to send an email notification to the portal administrator who can then send the registering person one of the predefined usernames and passwords. If the predefined list becomes exhausted then more usernames and passwords must be manually created and added to `users.sql` and the component restarted.

The registration form submission may be controlled by the use of the *reCAPTCHA* facility.

- *Local* SMTP service not configured.

Visitors to the `client` and/or `client-direct` interface who wish to register are directed to the “contact” page where it is expected there will be the relevant contact information.

- *reCAPTCHA*

A public and private key needs to be obtained from Google.

To use this feature there needs to be internet access as you need to call a Google internet service. (I'm not sure how flexible it is with intranet web addresses!).

Bespoke site-business functionality

Brief outline

If you intend to use the `client` portal to directly access your site's compound data then all the code to do so will need to be written.

To assist in this process there is the generic `site-business` component available in the *portal repository* which represents a fictional site compound database - this component mirrors the configuration required for your site's data to be passed to the generic `business-manager` processing via the `business-manager-api`.

Task responsibility

The primary objective of this activity involves a good deal of cooperation between the scientists and the relevant IT personnel in the organisation to determine how to extract the requisite assay data from the site databases.

The IT people involved will need to have some experience of Java software development (including *Spring*), as well as XML formats.

Technical Process

1. For the of the installation of the *generic portal components* each component's configuration requires the creation and assignment of *site-specific* files and values. Much of this process is based on deriving actual files from generic `sample.X` template files and then modifying their content to a site's requirements by following each component's specific installation instructions (e.g. as in the *installation* instructions).
2. For a bespoke `site-business` component there will be additional steps required (illustrated below) which will ultimately make use of a *Maven WarPath* plugin to overlay *site-specific* functionality over compiled and *Maven*-installed `business-manager` code.

The structure below and explanations thereafter (taken from the sample `site-business` component used for demonstration purposes), can be used to explain specifications and operations of generally required bespoke `site-business` functionality.

```

site-business/src/main/java/uk/ac/ox/cs/epsrc/site_business/business/selection/
↳maxresponse/CSDemoMaxRespStrategy.java          --+
|          |                                     |
↳+-----/CSMaxRespDataSelector.java             |          |
|          |                                     |          +---/SiteDataLoader.
↳java                                             |
|          |                                     |-----/dao/SiteDAOImpl.java
↳          |          |                         |
|          |          |                         |-----/request/validator/
↳SiteSimulationsRequestValidator.java            |          |
|          |          |                         |-----/service/
↳ExperimentalDataServiceImpl.java                 |          |
|          |          |                         |-----/SiteIdenifiers.java
↳          |          |                         |-- 1
|          |          |                         +-----/value/object/datarecord/
↳doserresponse/CSDoseResponseDataRecordVO.java   |          |
|          |          |                         |-----/
↳individual/CSIndividualDataRecordVO.java         |          |
|          |          |                         |          |
↳  |-----/CSManualPatchCaV12DataRecordVO.java   |          |
|          |          |                         |          |
↳  |-----/CSManualPatchHERGDataRecordVO.java     |          |
|          |          |                         |          |
↳  +-----/CSManualPatchNaV15DataRecordVO.java    |          |
|          |          |                         |          +-----/
↳summary/CSSummaryDataRecordVO.java              --+
|          |          |-----/resources/log4j.xml
|          |          |-----/META-INF/properties/.gitignore

```

(continues on next page)


```

|      |      |      |      +----/README
|      |      |      |      |--/spring/ctx/appCtx.business.site.xml
|      |      |      |      |-/config/appCtx.config.
→actionPotential.site.xml      --+
|      |      |      |      |      |--/appCtx.config.assays.site.
→xml                            |
|      |      |      |      |      |--/appCtx.config.cellModel.
→site.xml                      |
|      |      |      |      |      |--/appCtx.config.
→changeCheckers.site.xml      |-- 2
|      |      |      |      |      |--/appCtx.config.
→doseResponseFitting.site.xml |
|      |      |      |      |      |--/appCtx.config.ionChannels.
→site.xml                     |
|      |      |      |      |      |--/appCtx.config.
→pc50Evaluators.site.xml      |
|      |      |      |      |      +---/appCtx.config.site.xml   ]
→                                +-+
|      |      |      |      |      |-/data/site/appCtx.database.site.
→xml                          ----- 3
|      |      |      |      |      +-/integration/simulation/
→processing/experimental/appCtx.proc.site.experimental.xml +-+
|      |      |      |      |      |-----
→/qsar/appCtx.proc.site.qsar.xml          |-- 4
|      |      |      |      |      |-----+
→/screening/appCtx.proc.site.screening.xml  --+
|      |      |      |      |      +---/sql/site/site.embedded.sql    ]
→                                           --+___ 5
|      |      |      |      |      +-/site.mysql.sql                  ]
→                                           --+
|      |      |      |      |      +-/webapp/WEB-INF/spring/spring-site-config.xml
|      |      |      |      |      |-/properties/database/site/.gitignore
→                                           --+
|      |      |      |      |      |-/README                           ]
→                                           |-- 6
|      |      |      |      |      +-/sample.site.database.properties ]
→                                           --+
|      |      |      |      |      |-----/.gitignore                ]
→                                           --+___ 7
|      |      |      |      |      +----/sample.env.properties        ]
→                                           --+
→      +-/test/ ....
→                                           ----- 8

```

2 - Define CellMLs, Assays, Ion Channels, etc.

Note: A site's configuration files are ultimately derived from the XML files in the business-manager [sample configs](#) directory. Examples of how these templates are copied and modified to fit to a site's unique specifications is demonstrated in the site-business component [configs](#).

XML configuration files to define a site's unique setup.

1. `appCtx.config.actionPotential.site.xml` : ApPredict invocation settings.
 - a. Default frequencies and compound concentrations in situations where there is no *experimental data* available (ApPredict's `--pacing-freq` and `--plasma-concs` arg. values).
 - b. Default credible interval percentiles (ApPredict's `--credible-intervals` arg. value).² The extension of functionality of this argument appeared around end-May 2018 to improve on the previous use of an unmodifiable singular default credible interval percentile of 95%.
 - c. Default maximum pacing time (ApPredict's `--pacing-max-time` arg. value).
 - d. Default plasma concentration count (ApPredict's `--plasma-conc-count` arg. value).
 - e. Default plasma concentration log scale (ApPredict's `--plasma-conc-logscale` arg. value).
2. `appCtx.config.assays.site.xml` : Assay details.
 - a. Assay names.
 - b. *Assay levels*.¹ Assays in AP-Portal are considered to be hierarchical and a level value is assigned to each assay. Historically the hierarchy has reflected assay accuract, with the least accurate assay assigned a level value of 0 (zero), and increasingly more accurate assays assigned incremental level values.
 - c. *Assay groups*.¹ When different assays have similar characteristics they can be grouped, and as with **assay levels**, the grouping is hierarchical. Historically the assay groups levels assigned have mirrored the assay levels assigned, with the least accurate groups assigned a group level of 0 (zero) and increasingly more accurate assay groups assigned incremental group level values.
 - d. Assay *variability*.²
3. `appCtx.config.cellModel.site.xml` : CellML models.

The collection of CellML models contained in this configuration file **must** correspond to the CellML models which ApPredict is expecting. If you are uncertain which CellML models ApPredict has been built to work with, then invoke ApPredict at the command line with no arguments supplied, which should generate output containing something like the following :

```
*****
* ApPredict::Please provide some of these inputs:
*
* EITHER --model
*   options: 1 = Shannon, 2 = TenTusscher (06), 3 = Mahajan,
*             4 = Hund-Rudy, 5 = Grandi, 6 = O'Hara-Rudy 2011 (endo),
*             7 = Paci (ventricular), 8 = O'Hara-Rudy CiPA v1 2017 (endo)
*             9 = Faber-Rudy.
* OR --cellml <file>
```

It's unlikely that the CellML models data would need to change except in the specification of the `defaultModel` - i.e. the model which should appear selected by default in the UI (User Interface).

² See also [Variability Configuration](#).

¹ *assay levels* and *assay group* levels are used in *assay value inheritance*.

4. `appCtx.config.changeCheckers.site.xml` - Change checkers.

Change checkers should generally not require modification.

5. `appCtx.config.doseResponseFitting.site.xml` - *Dose-Response data* fitting preferences.

Only relevant if *dose-response data* fitting.

6. `appCtx.config.ionChannels.site.xml` - Active ion channels.

Collection of the actively tested ion channels.

7. `appCtx.config.pc50Evaluators.site.xml` - pIC50 evaluators.

pIC50 evaluation involves activating/deactivating and arranging evaluators according to site requirements. It is likely that the scientists will need to be consulted on which evaluators are to be used and their hierarchical order (0 (zero) being the first evaluator to seek a pIC50 value from). From experience it is likely that additional *site-specific* evaluators which also need to be implemented.

8. `appCtx.config.site.xml` - Various client options settings.

- a. `defaultForceReRun` - **use with care!**

- b. *Assay value inheritance* defaults.

- c. Information display levels.

- d. `defaultQuietPeriod` - Time (in minutes) following a simulation's run or re-run check in which another re-run check will not be undertaken. This is to avoid multiple concurrent requests to view a compound resulting in simulation re-runs.

3 - Connecting to the site's database

Connectivity to the site's database.

4 - Define Screening, QSAR, Experimental processing

Different sources of data to process, e.g. screening, QSAR, experimental data.

5, 6 - Ignore

(Sample site database structure)

7 - Define programming environment properties

Sample environment properties.

8 - Writing the Java unit testing classes.

Java test classes.

3.1.4 Generic AP-Portal Components

The following are *generic portal components*

- `app-manager` : ApPredict-invoking WS.
- `business-manager-api` : `site-business-developer` API (Application Programming Interface) to `business-manager` operations.
- `business-manager` : Generic ‘business’ processing WS.
- `client` : A web interface which talks to the `business-manager`.
- `client-direct` : A web interface which talks directly to the `app-manager`.
- `dose-response-jni` : Dose-Response fitting JNI (Java Native Interface).
- `dose-response-manager` : Dose-Response fitting WS. (which uses `dose-response-jni`).

The following are modularised components used when building some aspects of the portal.

- `client-parent` : *Maven* project build feature used by client components.
- `client-shared` : Shared functionality across the `client` and `client-direct` components.

The following is an example *site-specific* component containing fictional compound data and processing which interacts with the generic components for demonstration purposes.

- `site-business` : Example demonstrator of *site-specific* processing.

Component Installation

`app-manager`

`app-manager` contains code to invoke and monitor ApPredict.

`app-manager` was designed to enable the isolation of ApPredict invocation so that any client (e.g. a WS-invoking Perl script similar to `query_business_manager_WSS.pl` - not necessarily either of the AP-Portal `site-business` and/or `client-direct` components) could call it to run ApPredict. It was also isolated so that it could reside on hardware which was able to run many concurrent invocations, e.g. on a workstation with a lot of RAM and processors, without degrading the client’s response times.

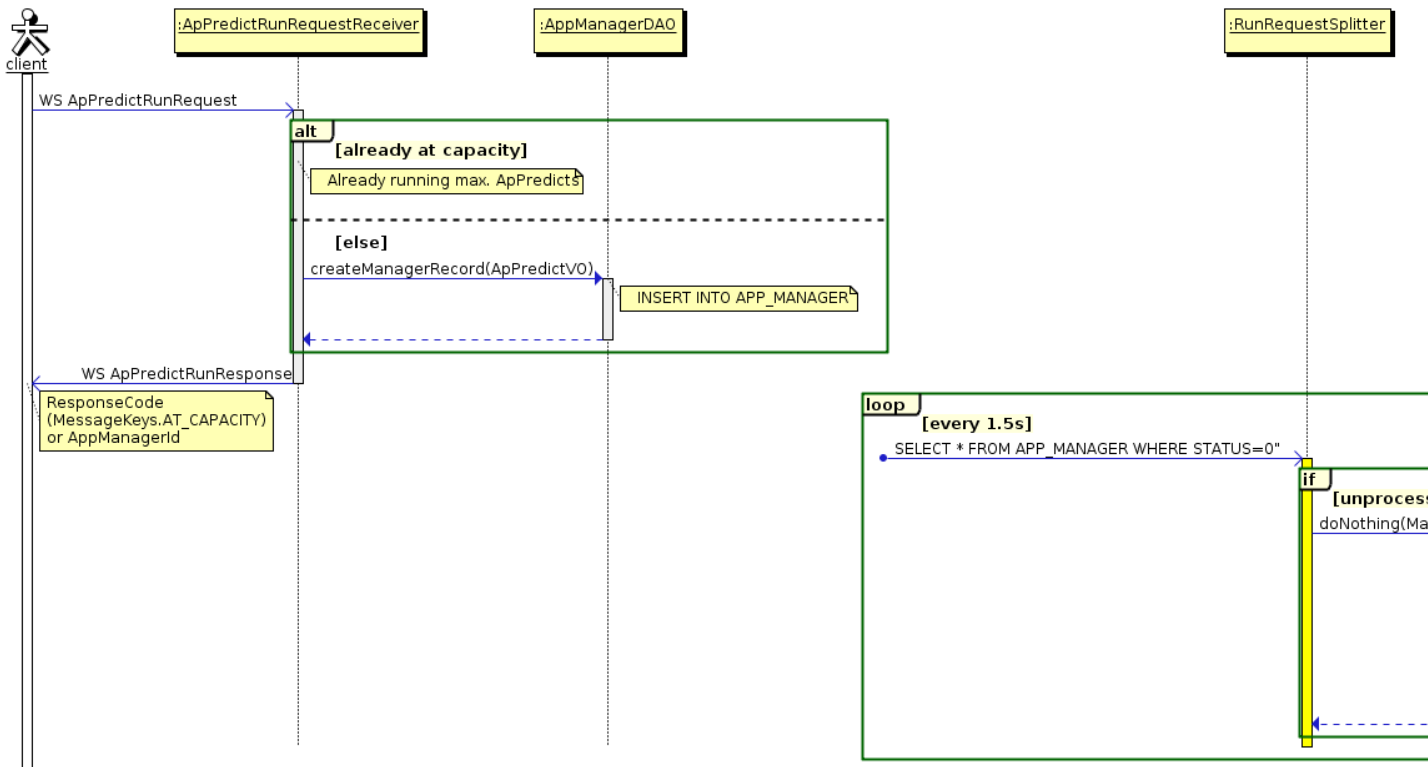
Below is a diagrammatic representation `app-manager` ApPredict invocation operations (see `tools/prepare_<appredict_invocation_mechanism>.sh`, `tools/local_runner.sh` and `tools/<appredict_invocation_mechanism>.sh` for more info).

Dependencies

- Maven (build)
- Java 7 or higher (build, deploy)
- Java Servlet Container, e.g. Apache *Tomcat* (deploy)
- Database (build, deploy)

Simplification of app-manager ApPredict invocation operations

The Spring Integration Workflow uses an inbound-channel-adapter to periodically query for new ApPredict invocation requests written to the database. New arrivals are split and sent along the Spring Integration channels (the diagram below misrepresents message passing along channels as returns from method calls!).



Initial installation

Download the project source and go to this component's root directory (i.e. [here](#)) and follow the steps below.

- Either ...
 1. Run `tools/one_time_copying.sh`
- ... or ...
 1. [T] (Technical content) Copy `sample.pom.xml` to `pom.xml`
 2. [T] Copy `src/main/resources/META-INF/spring/ctx/ws/sample.appCtx.ws.security-incoming.xml` to `src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-incoming.xml`
 3. [T] Copy `src/main/resources/META-INF/spring/ctx/ws/sample.appCtx.ws.security-outgoing.xml` to `src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml`
 4. [T] Copy `src/properties/database/sample.database.filter.properties` to `src/properties/database/database.filter.properties`
 5. [T] Copy `src/properties/sample.filter.properties` to `src/properties/filter.properties`
 6. [T] Copy `src/properties/sample.spring.properties` to `src/properties/spring.properties`
- ... then ...
 1. Follow database choice instructions.
 2. [T] Copy the bash shell scripts and RelaxNG schema files, e.g. ...
 1. `*.sh`
 2. `mathml2.rng`

... from `/app-manager/tools/` to your equivalent of *Tomcat's* `CATALINA_HOME`. Continue to read `tools/prepare_<appredict_invocation_mechanism>.sh`, `tools/local_runner.sh` and `tools/<appredict_invocation_mechanism>.sh` for further configuration instructions.
 3. [T] Create the directory `procdir` in your equivalent of *Tomcat's* `CATALINA_HOME`
- ... finally
 1. Edit each of the copied files according to your desired *configuration*.

Configuration

`pom.xml`

It is likely that there is a *site-specific* database driver (such as a *MySQL* driver) which needs adding to the `pom.xml` file.

`src/properties/filter.properties`

`app.soap.location=`

WS URL of this component, e.g. `'http://localhost:18380/app_manager-0.0.4/aws/'`

```
app.soap.wsdl=
```

This component's WSDL name, e.g. 'app_services.wsdl' The combination of `app.soap.location` and `app.soap.wsdl`, e.g. 'http://localhost:18380/app_manager-0.0.4/aws/app_services.wsdl' should reveal the SOAP web service when the component is running.

```
business.soap.location=
```

WS URL of site-business component, e.g. 'http://localhost:18080/site_business-0.0.2/bws/'

```
log.file=
```

Component log file location on disk, e.g. 'logs/app-manager.log'

```
log.level.app_manager=
```

Log level of this component, e.g. (trace|debug|info|warn|error|fatal).

```
log.level.non_app_manager
```

Log level of code from other libraries, e.g. (trace|debug|info|warn|error|fatal).

src/properties/spring.properties

```
base.dir=
```

Directory used as abase for ApPredict invocations, e.g. '/home/user/app_run/'

Warning: Assign cautiously! ApPredict is invoked in subdirectories of this directory and these subdirectories are created and destroyed (by [PostSimulationRunTidyUp.java](#)) as appropriate.

```
monitor.frequency=
```

Simulation process monitoring frequency (in seconds), e.g. '15'.

```
invocation.limit=
```

Maximum number of concurrent ApPredict invocations, e.g. '8'.

```
securement.app.username=
```

Component WSS (Web Services Security) username.

```
securement.app.password=
```

Component WSS password.

```
securement.business.username=
```

site-business WSS username.

```
securement.business.password=
```

site-business WSS password.

`src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-incoming.xml`

Generally no change necessary.

`src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml`

Generally no change necessary.

`src/properties/database/database.filter.properties`

Generally no change necessary.

`src/properties/database/dev.database.<deploy.db_vendor>.properties`

See also:

Database choice

`tools/prepare_<appredict_invocation_mechanism>.sh`

One of the following files will be your simulation preparation script of choice according to your local ApPredict installation.

1. `prepare_docker.sh`
2. `prepare_localfs.sh`
3. `prepare_singularity.sh`

Its role generally is to copy across (or symlink) the ApPredict invoker of choice file (see `tools/<appredict_invocation_mechanism>.sh`) into the directory in which the simulation will run (i.e. a subdirectory of `base.dir` (as defined in `src/properties/spring.properties`’ `base.dir`)), and to swap a couple of placeholder values therein.

The name `prepare.sh` is hardcoded into `ApPredictRunPreProcessor.java`, and therefore depending on how you intend to run ApPredict you will need to symlink one of above files to `prepare.sh`, as below ...

```
-rwx-----, 1 geoff geoff      4898 Dec 13  2016 local_runner.sh
-rwx-----, 1 geoff geoff      5293 Jan 22  2014 local_runner.sh.modified
drwxr-x---, 2 geoff geoff      4096 Feb 27 21:30 logs
-rw-rw-r--, 1 geoff geoff    99353 Jan 15  2018 mathml2.rng
-rwx-----, 1 geoff geoff     1576 Dec  8  2018 prepare_docker.sh
-rwx-----, 1 geoff geoff     1061 Mar 27  2018 prepare_localfs.sh
-rwx-----, 1 geoff geoff       913 May  4  2018 prepare_old.sh
lrwxrwxrwx, 1 geoff geoff        22 Feb 25 22:53 prepare.sh -> prepare_singularity.sh
-rwx-----, 1 geoff geoff     1644 Feb 25 23:16 prepare_singularity.sh
drwxr-x---, 2 geoff geoff      4096 Feb 25 23:39 procdir
-rw-r-----, 1 geoff geoff       104 Jan 22  2014 README.AP-preDICT.invocation
-rw-r-----, 1 geoff geoff       425 Feb 10  2014 README.openssl
-rwx-----, 1 geoff geoff       248 Nov 25 18:15 reset_all.sh
-rwx-----, 1 geoff geoff       889 Nov  4 17:12 reset.sh
drwxrwxr-x, 12 geoff geoff      4096 Dec  8  2018 save
-rwxr-x---, 1 geoff geoff       949 Feb 25 23:39 singularity.sh
```


`tools/local_runner.sh`

You shouldn't need to make any changes to this file.

The name `local_runner.sh` is hardcoded in `ApPredictInvoker.java`.

https://bitbucket.org/gef_work/ap_predict_online/src/master/app-manager/tools/local_runner.sh is the script that will be called by `ApPredictInvoker.java` and it will ...

1. `cd` to the *local* filesystem dir (as determined by the *base.dir* property and job identifier), then
2. run whichever `ApPredict.sh` wrapper script (as derived from `ApPredict.sh`, `Singularity.sh` or `Docker.sh`) which will in turn invoke the *ApPredict* binary.
3. It will also generate the *VRE_INFO* and *VRE_OUTPUT* files, the former contains general environment info, the latter contains the *stdout* and *stderr* from the invocation, both of which get persisted if the simulation runs successfully.

`tools/<appredict_invocation_mechanism>.sh`

One of the following files will be your *ApPredict* invoker of choice for your installation. The content of which you may need to modify according to your local *ApPredict* installation.

1. `ApPredict.sh`
2. `Singularity.sh`
3. `Docker.sh`

The name *ApPredict.sh* is hardcoded in `ApPredictInvoker.java`, and therefore you need to have symlinked your relevant `prepare_<appredict_invocation_mechanism>.sh` to `prepare.sh` (as explained in [tools/prepare_<appredict_invocation_mechanism>.sh](#)), such that the *ApPredict* invoker of choice wrapper script will be used (via an invocation of `local_runner.sh`).

`procdir`

`procdir` is used to deposit system process data into files which can be read by the application for the purpose of monitoring the system process state, i.e. to determine if *ApPredict* is still running. It is hardcoded in two places :

1. File `appCtx.int.processMonitoring.xml` has hardcoded (in `int-file:inbound-channel-adapter`) the directory `procdir` where the application expects system process data files to be deposited by the system at runtime upon *ApPredict* invocation.
2. `local_runner.sh` also has the same hardcoding.

Build

See also:

Database choice and *Spring profiles* for additional information.

`-Dspring.profiles.active=`

Options are currently `app_manager_(embedded|mysql|oracle10g)`.

Warning: At build time it is **important** to use the `-Dspring.profiles.active=app_manager_embedded` arg because during integration testing rubbish may be written to the database so it is important to use the embedded database during this process.

`-Ddeploy.db_vendor=`

The name of the database vendor, e.g. `mysql`, `postgres`, used in the intended deployment (**not build**) environment. The actual name used corresponds exactly to the `<deploy.db_vendor>` element of the file derived from `sample.database.spring.properties` during the installation process.

`-Ddeploy.env=`

The environment name, i.e. `dev`.

Example build instructions (illustrating use of *embedded* during the build process, and *mysql* in the deploy environment):

```
cd <ap_predict_online>/app-manager
mvn clean verify -Dspring.profiles.active=app_manager_embedded -Ddeploy.db_
↪ vendor=mysql -Ddeploy.env=dev
```

Deploy

If the build command has completed successfully the `webapp.war` file should be in the `target` directory. Copy this to the servlet container's relevant file, e.g for *Tomcat*, the `webapps` directory.

Run

See *start*.

Start-up problems

Property 'driverClassName' threw exception; nested exception is java.lang.IllegalStateException: Could not load JDBC driver class [appCtx.database.xml_unassigned]

Probably means that the `mvn` build command had the wrong combination of `-Ddeploy.db_vendor=` and/or `-Ddeploy.env=` values assigned. What generally happens is that the `pom.xml` `generate-resources` phase will concatenate two properties files into `src/main/resources/META-INF/properties/app_manager.properties` but if the wrong (or no) database properties file is referenced then the database properties will be missing.

No bean named 'appDataSource' is defined

It could be that wrong `JAVA_OPTS` value for `spring.profiles.active` is being set prior to the component being started, e.g. `JAVA_OPTS="-Dspring.profiles.active=app_manager_myysql"`.

Various notes

See JASYPT use.

business-manager

business-manager contains generic business processing code.

Dependencies

- *Maven* (build)
- Java 7 or higher (build, deploy)
- Java Servlet Container, e.g. Apache *Tomcat* (deploy)
- Database (build, deploy)
- A *Maven*-installed business-manager-api. See *business-manager-api*

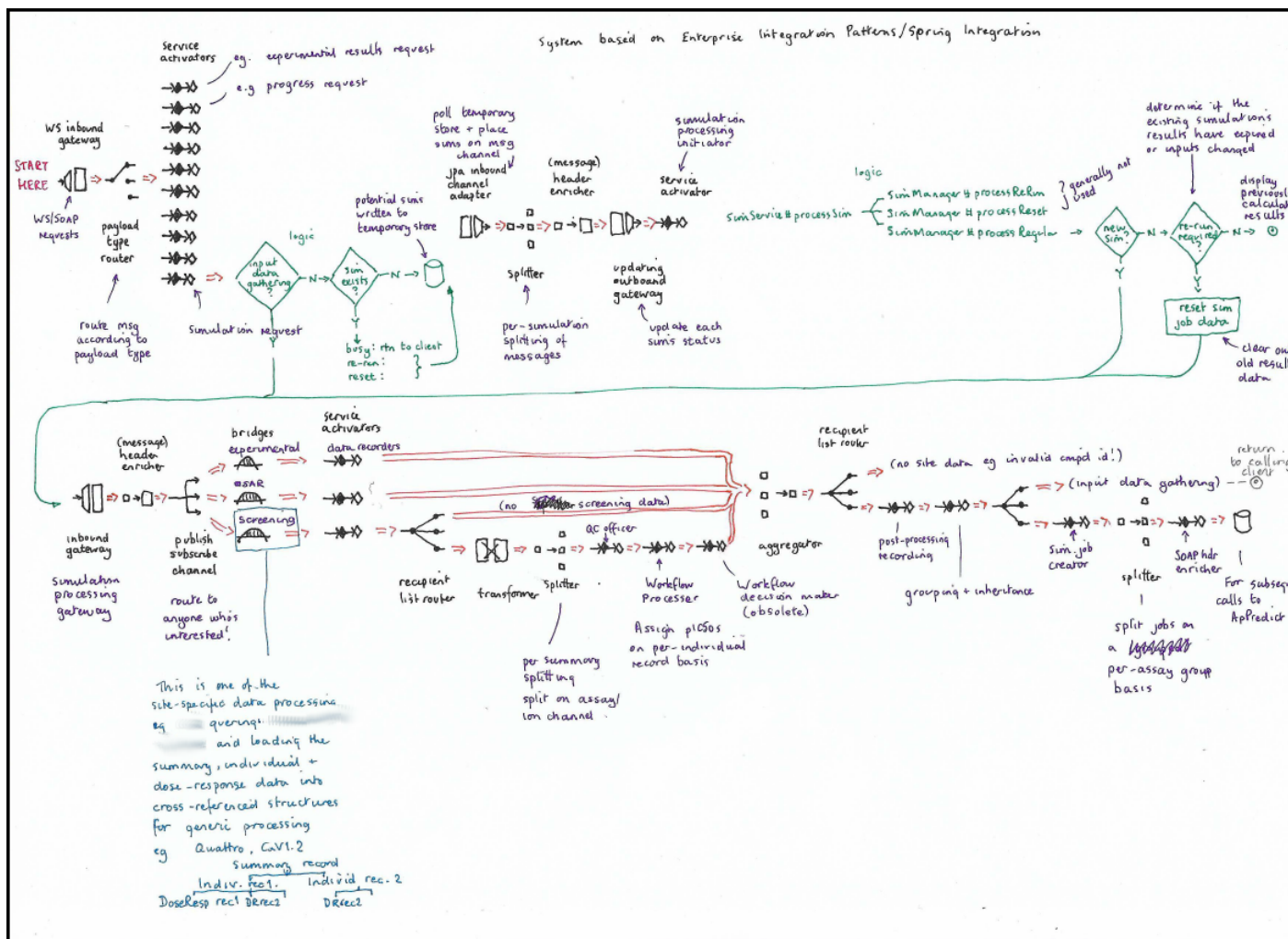
Initial installation

Download the project source and go to this component's root directory (i.e. [here](#)) and follow the steps below.

- Either ...
 1. Run `tools/one_time_copying.sh`
- ... or ...
 1. [T] Copy `sample.pom.xml` to `pom.xml`
 2. [T] Copy `src/main/resources/META-INF/spring/ctx/ws/sample.appCtx.ws.security-incoming.xml` to `src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-incoming.xml`
 3. [T] Copy `src/main/resources/META-INF/spring/ctx/ws/sample.appCtx.ws.security-outgoing.xml` to `src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml`
 4. [T] Copy `src/properties/database/sample.database.filter.properties` to `src/properties/database/database.filter.properties`
 5. [T] Copy `src/properties/sample.filter.properties` to `src/properties/filter.properties`
 6. [T] Copy `src/properties/sample.spring.properties` to `src/properties/spring.properties`
- ... then ...
 1. Follow database choice instructions.
- ... finally
 1. Edit each of the copied files according to your desired *configuration*.

Configuration

The illustration below depicts generic business-manager workflow :



`pom.xml`

It is likely that there is a *site-specific* database driver (such as a *MySQL* driver) which needs adding to the `pom.xml` file.

`src/properties/filter.properties`

```
app_manager.soap.location=
```

WS URL of app-manager component, e.g. `'http://localhost:18380/app_manager-0.0.4/aws/'`

```
business_manager.input_data_gathering.timeout=
```

Timeout (in milliseconds) whilst waiting for *input data gathering* requests to return. Depending on how responsive the site components are (e.g. web services), this may need to be a corresponding number of seconds.

Warning: If no response is received within the specified timeout the system is likely to consider it as an indication that no data is available, rather than issue a warning. (TODO: Verify this!)

```
business_manager.amq_transport_connector.host=
```

AMQ (ActiveMQ) Transport connector URI (Uniform Resource Identifier) host (see appCtx.jms.ActiveMQ.xml), e.g. '127.0.0.1'.

```
business_manager.amq_transport_connector.port=
```

AMQ Transport connector URI port (see appCtx.jms.ActiveMQ.xml), e.g. '61616'.

```
business_manager.request_processing.polling.filter=1000
```

Default simulation request processing polling period (in ms).

When incoming simulation requests are received a few first-phase “request processing” tests will take place immediately to determine if a simulation request should proceed to the second “simulation processing” phase. If it is determined that the simulation should proceed then it is persisted in the database and periodically (as determined by the value of this property) the database will be polled for simulations awaiting second-phase processing.

```
log.file.business_manager=
```

Component log file location on disk, e.g. 'logs/business-manager.log'

```
log.level.business_manager=
```

Component log level, e.g. (trace|debug|info|warn|error|fatal).

```
log.level.general=
```

Log level of code from other libraries, e.g. (trace|debug|info|warn|error|fatal).

src/properties/spring.properties

```
fdr.soap.location=
```

WS URL of dose-response-manager, e.g. 'http://127.0.0.1:18180/fdr_manager-0.0.1-SNAPSHOT/fws/'

```
app_manager.capacity_indicator=
```

app-manager operating at capacity indicator, e.g. 'appmanager.at_capacity' The value used corresponds to the client I18N (internationalisation) bundle identifier.

```
app_manager.progress_pfx=
```

app-manager default progress prefix, e.g. 'PROG: ' (@see app-manager's ProgressMonitor#PROGRESS_PREFIX?)

```
app_manager.status_date_format=
```

app-manager status date format, e.g. 'yyyy:MM:dd HH:mm:ss'

```
app_manager.default_saturation_level=
```

app-manager default saturation level (%) value, e.g. '0'

```
business_manager.request_processing.polling=@business_manager.request_processing.  
→polling.filter@
```

Generally no change necessary - value to use is derived from `filter.properties`.

```
fdr.default_coefficient=
```

dose-response-manager default Hill coefficient, e.g. '1'

```
securement.app.username=
```

app-manager WSS username.

```
securement.app.password=
```

app-manager WSS password.

```
securement.business.username=
```

Component WSS username.

```
securement.business.password=
```

Component WSS password.

`src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-incoming.xml`

Generally no change necessary.

`src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml`

Generally no change necessary.

`src/properties/database/database.filter.properties`

```
business_manager.database.queryTimeout=
```

Hibernate (`javax.persistence.query.timeout`) query timeout (in milliseconds), e.g. '1'.

```
business_manager.database.hbm2ddl=
```

Hibernate `hbm2ddl` schema DDL options e.g. (`validate|update|create|create-drop`).

`src/properties/database/dev.database.<deploy.db_vendor>.properties`

See also:

Database choice

Build

See also:

Database choice and *Spring profiles* for additional information.

`-Dspring.profiles.active=`

Options are currently `business_manager__` (`embedded` | `mysql` | `oracle10g`).

Warning: At build time it is **important** to use the `-Dspring.profiles.active=business_manager_embedded` arg because during integration testing rubbish may be written to the database so it is important to use the embedded database during this process.

`-Ddeploy.db_vendor=`

The name of the database vendor, e.g. `mysql`, `postgres`, used in the intended deployment (**not build**) environment. The actual name used corresponds exactly to the `<deploy.db_vendor>` element of the file derived from `sample.database.spring.properties` during the installation process.

Example build instructions :

```
cd <ap_predict_online>/business-manager
#export MAVEN_OPTS="-Xms512m -Xmx2g -XX:PermSize=2048m -XX:MaxPermSize=2048m"
mvn clean install -Dspring.profiles.active=business_manager_embedded -Ddeploy.db_
↪vendor=mysql
```

Note: Unlike in other components the `-Ddeploy.env` is hardcoded as `dev`.

Deploy

This component when built should not be deployed into a servlet container, instead it is *Maven*-installed in the *local Maven* repository and then a subsequent build of `site-business` retrieves the built `.war` file and uses it in the `site-business` build process.

Run

See *Deploy*

Start-up problems

See *Deploy*

Done that! What's next?

Conventionally the next step, having built and *Maven*-installed `business-manager`, would be to build a `site-business`. This is because a change to `business-manager` operations and/or configurations will only be visible once a `site-business` is subsequently built and deployed (as the build of a `site-business` would use the latest installed `business-manager`).

`business-manager-api`

`business-manager-api` contains the API to `business-manager` operations, i.e. it allows a/your `site-business` to pass data to the generic `business-manager` operations.

Dependencies

- *Maven* (build)
- Java 7 or higher (build, deploy)

Initial installation

Download the project source and go to this component's root directory (i.e. [here](#)) and follow the steps below.

Configuration

No configuration necessary.

Build

Example build instructions :

```
cd <ap_predict_online>/business-manager-api
mvn clean install
```

Deploy

This component when built should not be deployed into a servlet container, instead it is *Maven*-installed in the *local Maven* repository and then a subsequent build of `business-manager` or `site-business` retrieves the built `.war` file and uses it.

Run

See *Deploy*

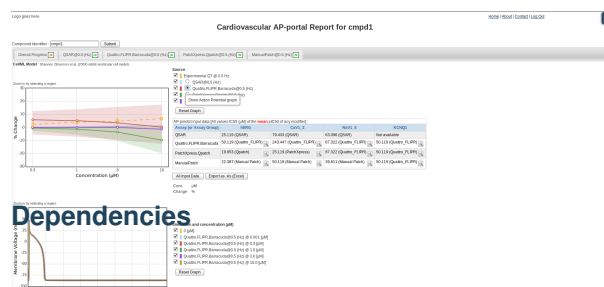
Start-up problems

See *Deploy*

Done that! What's next?

Conventionally the next step, having built and *Maven*-installed *business-manager-api*, would be to build *business-manager* and then a *site-business*. This is because both *business-manager* and *site-business* are dependent on *business-manager-api* and need to be kept in-sync.

client



client is an SPA (Single-Page Application) providing the web interface to *site-business*.

The *client* interface is intended for use at sites that intend to configure the AP-Portal to directly query their compound databases.

- *Maven* (build)
- Java 7 ¹ or higher (build, deploy)
- Java Servlet Container, e.g. Apache *Tomcat* (deploy)
- Database (build, deploy)
- A *Maven*-installed *client-parent* and

client-shared. See *client-parent* and *client-shared*

General topics

- Database
- Internationalisation
- *User Authentication and Authorisation*.
- *User Registration*.

Initial installation

Download the project source and go to this component's root directory (i.e. [here](#)) and follow the steps below.

- Either ...
- 1. Run `tools/one_time_copying.sh`
- ... or ...
- 1. [T] Copy `sample.pom.xml` to `pom.xml`
- 2. [V] (Visualisation content) Copy `src/main/resources/bundle/sample.site.properties` to `src/main/resources/bundle/site.properties`

¹ Since early 2019.

3. [S] (Scientific content) Copy `src/main/resources/META-INF/spring/ctx/config/sample.appCtx.config.site.xml` to `src/main/resources/META-INF/spring/ctx/config/appCtx.config.site.xml`
4. [T] Copy `src/main/resources/META-INF/spring/ctx/ws/sample.appCtx.ws.security-outgoing.xml` to `src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml`
5. [T] Copy `src/properties/database/sample.database.filter.properties` to `src/properties/database/database.filter.properties`
6. [T] Copy `src/properties/database/sample.database.spring.properties` to `src/properties/database/dev.database.embedded.properties`
7. See Database choice
8. [T] Copy `src/properties/sample.filter.properties` to `src/properties/filter.properties`
9. [T] Copy `src/properties/sample.spring.properties` to `src/properties/spring.properties`

If you intend to have a client-specific “*prepopulated user authentication*” *user authentication mechanism* database then also do the following :

1. [T] Copy `src/main/resources/META-INF/data/spring-security/local/sample.users.sql` to `src/main/resources/META-INF/data/spring-security/local/users.sql`
 - ... finally
1. Edit each of the copied files according to your desired *configuration*.

Configuration

`pom.xml`

It may necessary in some circumstances, e.g. if there is *site-specific* access control code being overlayed into the client `src/main/java` directory, or if using a *MySQL* driver, to adapt the `pom.xml` file for *Maven*-building the component.

`src/main/resources/bundle/site.properties`

`site.compound_identifier=`

Adjust the prompt message for compound identifiers (if necessary).

If the site is to be internationalised create all the corresponding language properties files, e.g. `src/main/resources/site_(es,zh).properties`.

`src/main/resources/META-INF/data/spring-security/local/users.sql`

Adjust this according to your expected usage requirements :

- If you are using a *local* client-specific “*prepopulated user authentication*” mechanism for authentication and authorisation, this file is required. Please also check the following **Note**.

- If you are using a “*bespoke user authentication*” mechanism, or if “*prepopulated user authentication*” users are shared between `client` and `client-direct` installations (and hence defined in `client-shared`), then this file is not required.

Note: If there is a new registration request the requesting user’s details are not automatically written to the user database. New users must be manually added to this `users.sql` file and the system restarted – for this reason it is better to create a number of unissued usernames and passwords (commensurate with the total anticipated usage) and then distribute them to new registrations as they arrive.

`src/main/resources/META-INF/spring/ctx/config/appCtx.config.site.xml`

`c50Units`

Preferred units for displaying inhibitory concentration, e.g. IC50 [nM], IC50 [μM], pIC50. Cut-and-paste your preferred unit to be the top/first `<entry />` value.

`configuration -> inputValueDisplay`

Choose how to derive the mean of pIC50 APredict invocation input values.

Set the value of the `ref=""` attribute to one of the following :

1. `INPUT_VALUE_DISPLAY_ANY_MODIFIER` Display the mean of all pIC50 values, irrespective of modifier (i.e. include ‘=’, ‘<’, ‘>’).
2. `INPUT_VALUE_DISPLAY_EQUALITY_MODIFIER` Display the mean of pIC50 values only where there was an equality modifier (i.e. use only ‘=’).

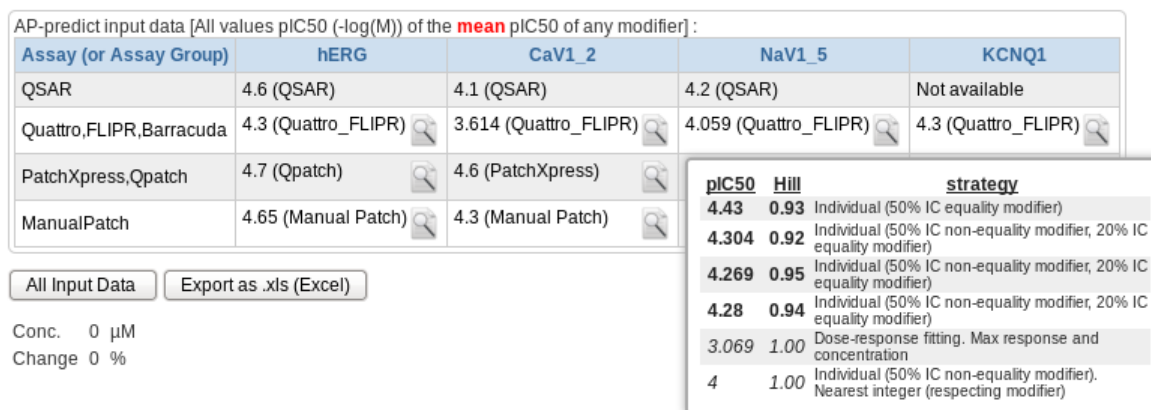


Fig. 1: Use of `INPUT_VALUE_DISPLAY_ANY_MODIFIER`.

`src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml`

Generally no change necessary.

`src/properties/database/database.filter.properties`

```
client_direct.database.queryTimeout=
```

Data query timeout (in milliseconds). Shouldn't need to be higher than 200?

`src/properties/database/dev.database.<deploy.db_vendor>.properties`

See also:

Database choice

`src/properties/filter.properties`

```
log.file.client=
```

Component log file location on disk, e.g. 'logs/client.log'

```
log.level.client=
```

Component log level, e.g. (trace|debug|info|warn|error|fatal).

```
log.level.general=
```

Log level of code from other libraries, e.g. (trace|debug|info|warn|error|fatal).

```
business_services.ws.url=
```

WS URL of site-business component, e.g. 'http://localhost:18080/site_business-0.0.2/bws/'

`src/properties/spring.properties`

```
recaptcha.private.key=
```

reCAPTCHA private key.

```
recaptcha.public.key=
```

reCAPTCHA public key. If this field is left empty it will be assumed that no reCAPTCHA is available!

```
securement.business.username=
```

site-business WSS username.

```
securement.business.password=
```

site-business WSS password.

Build

See also:

Database choice and *Spring profiles* for additional information.

`-Dspring.profiles.active=`

Options are currently `client-shared_(embedded|mysql|oracle10g)`, `client-shared_(bespoke|prepopulated)`.

Warning: At build time it is **important** to include the `client-shared_embedded` spring profile during building because during integration testing rubbish may be written to the database so it is important to use the embedded database during the build process.

`-Ddeploy.db_vendor=`

The name of the database vendor, e.g. `mysql`, `postgres`, used in the intended deployment (**not build**) environment. The actual name used corresponds exactly to the `<deploy.db_vendor>` element of the file derived from `sample.database.spring.properties` during the installation process.

`-Ddeploy.env=`

The environment name, i.e. `dev`.

Example build instructions (illustrating use of *embedded* during the build process, and *mysql* in the deploy environment):

```
cd <ap_predict_online>/client
mvn clean verify -Dspring.profiles.active=client-shared_embedded,client-shared_
↪prepopulated -Ddeploy.db_vendor=mysql -Ddeploy.env=dev
```

Deploy

If the *Build* command has completed successfully the webapp `.war` file should be in the target directory. Copy this to the servlet container's relevant file, e.g. for *Tomcat*, the `webapps` directory.

Run

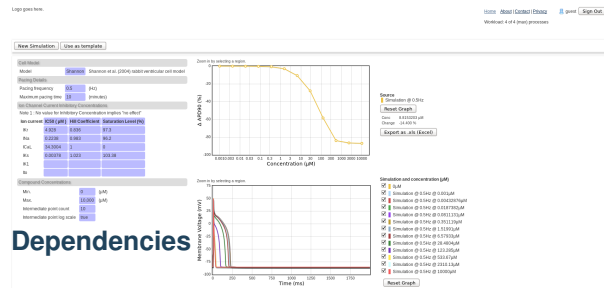
See *start*.

Start-up problems

See also:

Log file troubleshooting.

client-direct



client-direct provides the web interface to the app-manager component and uses a number of different web pages, i.e. it's not an SPA.

The client-direct interface is intended for running simulations using manually input values, e.g. pIC50, Hill Coefficients.

- *Maven* (build)
- Java 7 ¹ or higher (build, deploy)
- Java Servlet Container, e.g. Apache *Tomcat* (deploy)
- Database (build, deploy)
- A *Maven*-installed client-parent and client-shared. See *client-parent* and *client-shared*

General topics

- Database
- Internationalisation
- *User Authentication and Authorisation.*
- *User Registration.*

Initial installation

Download the project source and go to this component's root directory (i.e. [here](#)) and follow the steps below.

- Either ...
- 1. Run `tools/one_time_copying.sh`
- ... or ...
- 1. [T] Copy `sample.pom.xml` to `pom.xml`
- 2. [S] Copy `src/main/resources/META-INF/spring/ctx/config/sample.appCtx.config.cellModels.site.xml` to `src/main/resources/META-INF/spring/ctx/config/appCtx.config.cellModels.site.xml`
- 3. [S] Copy `src/main/resources/META-INF/spring/ctx/config/sample.appCtx.config.site.xml` to `src/main/resources/META-INF/spring/ctx/config/appCtx.config.site.xml`
- 4. [T] Copy `src/main/resources/META-INF/spring/ctx/ws/sample.appCtx.ws.security-outgoing.xml` to `src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml`

¹ Since early 2019.

5. [V] Copy `src/main/webapp/resources/js/site/sample.site.js` to `src/main/webapp/resources/css/site/site.js`
6. [T] Copy `src/properties/database/sample.database.filter.properties` to `src/properties/database/database.filter.properties`
7. [T] Copy `src/properties/database/sample.database.spring.properties` to `src/properties/database/dev.database.embedded.properties`
8. See Database choice
9. [T] Copy `src/properties/sample.filter.properties` to `src/properties/filter.properties`
10. [T] Copy `src/properties/sample.spring.properties` to `src/properties/spring.properties`

If you intend to have a `client-direct-specific` “*prepopulated user authentication*” *user authentication mechanism* database then also do the following :

1. [T] Copy `src/main/resources/META-INF/data/spring-security/local/sample.users.sql` to `src/main/resources/META-INF/data/spring-security/local/users.sql`
 - ... finally
1. Edit each of the copied files according to your desired *configuration*.

Configuration

`pom.xml`

It may necessary in some circumstances, e.g. if there is *site-specific* access control code being overlayed into the `client-direct` `src/main/java` directory, or if using a *MySQL* driver, to adapt the `pom.xml` file for *Maven*-building the component.

`src/main/resources/META-INF/data/spring-security/local/users.sql`

Adjust this according to your expected usage requirements.

- If you are using a *local* `client-direct-specific` “*prepopulated user authentication*” mechanism for authentication and authorisation, this file is required. Please also check the following **Note**.
- If you are using a “*bespoke user authentication*” mechanism, or if “*prepopulated user authentication*” users are shared between `client` and `client-direct` installations (and hence defined in `client-shared`), then this file is not required.

Note: If there is a new registration request the requesting user’s details are not automatically written to the user database. New users must be manually added to this `users.sql` file and the system restarted – for this reason it is better to create a number of unissued usernames and passwords (commensurate with the total anticipated usage) and then distribute them to new registrations as they arrive.

`src/main/resources/META-INF/spring/ctx/config/appCtx.config.cellModels.site.xml`

Collection of ion channel models which the user can choose from.

Probably the only modifications likely are :

1. To comment out any ion channel model(s) which is/are not relevant to the portal users, and/or,
2. To assign a default ion channel model to use by way of assigning a *single* ion channel model with a `defaultModel` arg with an attribute value of `true`.

Warning: These ion channel models are built into ApPredict and the detail **MUST** be identical to the values which ApPredict expects, particularly the `identifier` and `name` values.

`src/main/resources/META-INF/spring/ctx/config/appCtx.config.site.xml`

`c50Units`

Preferred units for displaying inhibitory concentration, e.g. IC50 [nM], IC50 [μM], pIC50. Cut-and-paste your preferred unit to be the top/first `<entry />` value.

`configuration -> recommendedPlasmaConcMax`

Default *recommended* value for maximum plasma concentration (μM) value. Users can enter values higher than the recommended maximum if they wish, i.e. this is not an enforced limit.

`configuration -> plasmaConcMin`

Minimum plasma concentration (μM). Users cannot enter values lower than this, i.e. it is an enforced limit.

`configuration -> spreads`

Define the default variability values for specified ion channels.

See also:

[Variability Configuration](#)

`src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml`

Generally no change necessary.

`src/main/webapp/resources/css/site/site.js`

Generally no change necessary.

`src/properties/database/database.filter.properties`

`client.database.queryTimeout=`

Data query timeout (in milliseconds). Shouldn't need to be higher than 200.

`src/properties/database/dev.database.<deploy.db_vendor>.properties`

See also:

Database choice

`src/properties/filter.properties`

`app_manager.soap.location=`

WS URL of app-manager component, e.g. `'http://localhost:18380/app_manager-0.0.4/aws/'`

See also:

For the following email-related properties take a look at the [JavaMail API documentation](#)

`log.file.client_direct=`

Component log file location on disk, e.g. `'logs/client-direct.log'`.

`log.level.client_direct=`

Component log level, e.g. `(trace|debug|info|warn|error|fatal)`.

`log.level.general=`

Log level of code from other libraries, e.g. `(trace|debug|info|warn|error|fatal)`.

`src/properties/spring.properties`

`recaptcha.private.key=`

reCAPTCHA private key.

`recaptcha.public.key=`

reCAPTCHA public key. If this field is left empty it will be assumed that no reCAPTCHA is available!

`securement.app.username=`

app-manager WSS username.

`securement.app.password=`

app-manager WSS password.

Build

See also:

Database choice and [Spring profiles](#) for additional information.

`-Dspring.profiles.active=`

Options are currently `client-shared_(embedded|mysql|oracle10g)`, `client-shared_(bespoke|prepopulated)`.

Warning: At build time it is **important** to include the `client-shared_embedded` spring profile during building because during integration testing rubbish may be written to the database so it is important to use the embedded database during the build process.

`-Ddeploy.db_vendor=`

The name of the database vendor, e.g. `mysql`, `postgres`, used in the intended deployment (**not build**) environment. The actual name used corresponds exactly to the `<deploy.db_vendor>` element of the file derived from `sample.database.spring.properties` during the installation process.

`-Ddeploy.env=`

The environment name, i.e. `dev`.

Example build instructions (illustrating use of *embedded* during the build process, and *mysql* in the deploy environment):

```
cd <ap_predict_online>/client-direct
mvn clean verify -Dspring.profiles.active=client-shared_embedded,client-shared_
  ↪prepopulated -Ddeploy.db_vendor=mysql -Ddeploy.env=dev
```

Deploy

If the *Build* command has completed successfully the webapp `.war` file should be in the target directory. Copy this to the servlet container's relevant file, e.g. for *Tomcat*, the `webapps` directory.

Run

See *start*.

Start-up problems

See also:

Log file troubleshooting.

`client-parent`

`client-parent` contains the *Maven parent pom* for determining the shared build configurations of the client components.

Dependencies

- *Maven* (build)
- Java 7 or higher (build, deploy)

Initial installation

Download the project source and go to this component's root directory (i.e. [here](#)) and follow the steps below.

Configuration

No configuration necessary.

Build

Example build instructions :

```
cd <ap_predict_online>/client-parent
mvn --non-recursive clean install
```

The optional `--non-recursive` instruction prevents the sub-projects from automatically being built.

Deploy

This component when built should not be deployed into a servlet container, instead it is *Maven*-installed in the *local Maven* repository and then a subsequent build of `client-shared` retrieves the built `.pom` files and uses it.

Run

See *Deploy*

Start-up problems

See *Deploy*

Done that! What's next?

Conventionally the next step, having built and *Maven*-installed `client-parent`, would be to build `client-shared` and thereafter `client` and/or `client-direct`. This is because all of `client-shared`, `client` and `client-direct` are dependent on `client-parent` and need to be kept in-sync.

client-shared

client-shared is some of the shared operations and configurations between the client and client-direct components.

See options for customisable *User Authentication and Authorisation*.

See options for customisable *user registration*.

Dependencies

- *Maven* (build)
- Java 7 ¹ or higher (build, deploy)

Initial installation

Download the project source and go to this component's root directory (i.e. `client-shared`) and follow the steps below.

- Either ...
 1. Run `tools/one_time_copying.sh`
- ... or ...
 1. [T] Copy `sample.pom.xml` to `pom.xml`
 2. [T] Copy `src/main/resources/META-INF/data/spring-security/local/sample.users.sql` to `src/main/resources/META-INF/data/spring-security/local/users.sql`
 3. [T] Copy `src/main/resources/META-INF/resources/resources/css/site/sample.client-shared-site.css` to `src/main/resources/META-INF/resources/resources/css/site/client-shared-site.css`
 4. [T] Copy `src/main/resources/META-INF/resources/WEB-INF/spring/ctx/config/sample.appCtx.features.xml` to `src/main/resources/META-INF/resources/WEB-INF/spring/ctx/config/appCtx.features.xml`
 5. [T] Copy `src/main/resources/META-INF/resources/WEB-INF/tiles/layout/common/sample.logo.jsp` to `src/main/resources/META-INF/resources/WEB-INF/tiles/layout/common/logo.jsp`
 6. [T] Copy `src/main/resources/META-INF/resources/WEB-INF/tiles/layout/common/sample.logout.jsp` to `src/main/resources/META-INF/resources/WEB-INF/tiles/layout/common/logout.jsp`
 7. [T] Copy `src/main/resources/META-INF/resources/WEB-INF/tiles/layout/contact/sample.contact.jsp` to `src/main/resources/META-INF/resources/WEB-INF/tiles/layout/contact/contact.jsp`
 8. [S] Copy `src/main/resources/WEB-INF/spring/authn/sample.appCtx.bespoke.xml` to `src/main/resources/WEB-INF/spring/authn/appCtx.bespoke.xml`
 9. [S] Copy `src/main/resources/WEB-INF/spring/authn/sample.appCtx.sitePreauthFilter.xml` to `src/main/resources/WEB-INF/spring/authn/appCtx.sitePreauthFilter.xml`

¹ Since early 2019.

10. [S] Copy `src/main/resources/WEB-INF/spring/sample.root-context.site.xml` to `src/main/resources/WEB-INF/spring/root-context.site.xml`
11. [T] Copy `src/properties/sample.cs-filter.properties` to `src/properties/cs-filter.properties`
 - ... finally
1. Edit each of the copied files according to your desired *configuration*.

Configuration

`pom.xml`

It may necessary in some circumstances, e.g. if there is *site-specific* access control code being overlayed into the client `src/main/java` directory, or if using a *MySQL* driver, to adapt the `pom.xml` file for *Maven*-building the component.

`src/main/resources/META-INF/data/spring-security/local/users.sql`

Adjust this according to your expected usage requirements :

- If you are using a *local* client- and/or client-direct-specific “*prepopulated user authentication*” mechanism for authentication and authorisation, this file is required. Please also check the following **Note**.
- If you are using a “*bespoke user authentication*” mechanism then this file is not required.

Note: If there is a new registration request the requesting user’s details are not automatically written to the user database. New users must be manually added to this `users.sql` file and the system restarted – for this reason it is better to create a number of unissued usernames and passwords (commensurate with the total anticipated usage) and then distribute them to new registrations as they arrive.

`src/main/resources/META-INF/resources/resources/css/site/client-shared-site.css`

Generally no change necessary.

`src/main/resources/META-INF/resources/WEB-INF/spring/ctx/config/appCtx.features.xml`

Adjust the visibility of certain portal features (e.g. entering ion channel spread values) to certain groups of users as determined by their “*Role*”.

See also:

Authentication Mechanisms

`src/main/resources/META-INF/resources/WEB-INF/spring/ctx/appCtx.multipart.xml`

Generally no change necessary.

- `filterMultipartResolver` controls the maximum file size which can be uploaded.
- `fileStorageCellML` and `fileStoragePK` control the enforcement of UTF-8 encoding file content.

See also:

[*src/properties/cs-filter.properties*](#) for assignment of `multipart.fileupload.maxinmemorysize` and `multipart.fileupload.maxuploadsize`.

See also:

MySQL's `max_allowed_packet`, which may need to be set to perhaps 40Mb or more, depending on PK file data content size, and, `uk.ac.ox.cs.compbio.client_shared.entity.PortalFile#MAX_SIZE`

`src/main/resources/META-INF/resources/WEB-INF/tiles/layout/common/logo.jsp`

Adjust the HTML according to the desired appearance of the top-left of the page which is usually where the logo is found.

The example `logo.jsp` content below places the fictional company logo in the top left of the display by referencing the corresponding logo image placed in the `src/main/resources/META-INF/resources/resources/img/site/` directory prior to build.

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
" alt="Company logo" /
↪>
```

`src/main/resources/META-INF/resources/WEB-INF/tiles/layout/common/logout.jsp`

Adjust according to your logging out mechanism.

`src/main/resources/META-INF/resources/WEB-INF/tiles/layout/contact/contact.jsp`

Adjust the HTML to provide contact details of the relevant people.

An example `contact.jsp` content may be as follows.

```
Click the following links to email support :

<p>
  Scientific support : <a href="mailto:ap-portal-scientific@company.com">click here</
↪a>
</p>
<p>
  Technical support : <a href="mailto:ap-portal-technical@company.com">click here</a>
</p>
```

`src/main/resources/WEB-INF/spring/authn/appCtx.bespoke.xml`

Generally no change necessary.

`src/main/resources/WEB-INF/spring/authn/appCtx.sitePreauthFilter.xml`

Needs to be adjusted if adopting the “*bespoke user authentication*” mechanism.

`src/main/resources/WEB-INF/spring/root-context.site.xml`

Generally no change necessary.

`src/properties/cs-filter.properties`

See also:

For the following email-related properties take a look at the [JavaMail API documentation](#)

`mail.smtp.host=`

New registration email mail server SMTP host, e.g. ‘smtp.company.com’.

`mail.smtp.port=`

New registration email mail server port, e.g. ‘25’ or ‘587’.

Warning: Always assign an integer value here, even if no SMTP service is available.

`mail.smtp.username=`

New registration email mail server user name.

`mail.smtp.password=`

New registration email mail server user’s password.

`mail.regn.from=`

New registration email “From” email address, e.g. ‘noreply-portal-registrations@company.com’.

`mail.regn.to=`

New registration email “To” email address, e.g. ‘portal-registrations@company.com’.

`mail.regn.subject=`

New registration email subject title, e.g. ‘New registration’.

`mail.transport.protocol=`

New registration email mail transport protocol (see `appCtx.mail.xml`), e.g. ‘smtp’.

`mail.smtp.auth=`

New registration email SMTP authentication required (see `appCtx.mail.xml`), e.g. `(true|false)`.

```
mail.smtp.connectiontimeout=
```

New registration email SMTP connection timeout (i.e. time to respond once connected) in milliseconds (see `appCtx.mail.xml`), e.g. '2000'.

```
mail.smtp.timeout=
```

New registration email SMTP timeout (to establish a connection) in milliseconds (see `appCtx.mail.xml`), e.g. '2000'.

```
mail.smtp.starttls.enable=
```

New registration email SMTP connection instruction to start TLS (Transport Level Security) (see `appCtx.mail.xml`), e.g. `(true|false)`.

```
mail.smtp.ssl.enable=
```

New registration email SMTP connection instruction to start TLS (see `appCtx.mail.xml`), e.g. `(true|false)`.

```
mail.debug=
```

New registration email debugging switch (see `appCtx.mail.xml`), e.g. `(true|false)`.

```
multipart.fileupload.maxinmemorysize=
```

From the [Spring docs](#) :

Set the maximum allowed size (in bytes) before uploads are written to disk. Uploaded files will still be received past this amount, but they will not be stored in memory. Default is 10240, according to Commons FileUpload.

```
multipart.fileupload.maxuploadsize=
```

From the docs [here](#) :

Set the maximum allowed size (in bytes) before uploads are refused. -1 indicates no limit (the default).

```
log.file.client_shared=
```

Component log file location on disk, e.g. 'logs/client-shared.log'.

```
log.level.client_shared=
```

Component log level, e.g. `(trace|debug|info|warn|error|fatal)`.

```
log.level.general=
```

Log level of code from other libraries, e.g. `(trace|debug|info|warn|error|fatal)`.

Build

Example build instructions :

```
cd <ap_predict_online>/client-shared
mvn clean install
```


Deploy

This component when built should not be deployed into a servlet container, instead it is *Maven*-installed in the *local Maven* repository and then a subsequent build of `client` and/or `client-direct` retrieves the built files and uses them.

Run

See *Deploy*

Start-up problems

See *Deploy*

Done that! What's next?

Conventionally the next step, having built and *Maven*-installed `client-shared`, would be to build `client` and/or `client-direct`. This is because both of `client` and `client-direct` are dependent on `client-shared` (and `client-parent`) and need to be kept in-sync.

dose-response-jni

`dose-response-jni` provides the `libfittingdoseresponse.so` and `dose-response-fitter.jar` files which are used by `dose-response-manager`. The original *dose-response data* fitting C++ source code written by Kylie Beattie can be found in *Kylie's repository*.

Note: TODO: Test and explain that if there is no *dose-response data* then it is not necessary to build/deploy `dose-response-jni`, `dose-response-manager`!

Dependencies

- Java 7 or higher (build)
- `boost-devel` (or a non-package-installed *local* equivalent) (build)
- `g++` (build)
- `make` (build)
- `libjson` (build)

Initial installation

Download the project source and go to this component's root directory (i.e. [here](#)) and follow the steps below.

1. [T] Copy `sample.makefile` to `makefile`

Configuration

makefile

This is the only file which should need to change.

```
JAVA_HOME=
```

Adjust this according to wherever your Java `jar`, `java`, `javac` and `javah` binaries are located (try which `javah`), e.g. `/usr`. NOTE: The binaries may not all be located in the same directory, in which case manually adjust each `$(JAVA_HOME)` occurrence.

```
BOOST_HOME=
```

Specify a value if your Boost include directory is not in a default path for searching, e.g. `/home/me/myincludes/boost`. NOTE: If the include is in a default path for searching then you can remove the `-I$(BOOST_HOME)` from the `CPPFLAGS` line.

```
JSON_HOME=
```

Specify a value if your libjson include and lib directories are not in default paths for searching, e.g. `/home/me/mylibjson`. NOTE: If the libjson include and lib directories are in default paths for searching then you can remove the references to `$(JSON_HOME)` from the file.

Build

Example build instructions :

```
cd <ap_predict_online>/dose-response-jni
make
```

Deploy

If the `make` command has completed successfully the files `dose-response-fitter.jar` and `libfittingdoseresponse.so` should be in the target directory. Copy these two files to the lib directory of `dose-response-manager`.

Done that! What's next?

Conventionally it will be *dose-response-manager*.

dose-response-manager

`dose-response-manager` provides the WS front-end to the dose-response fitting functionality.

Note: TODO: Test and explain that if there is no *dose-response data* then it is not necessary to build/deploy `dose-response-jni`, `dose-response-manager`!

Dependencies

- *Maven* (build)
- Java 7 or higher (build, deploy)
- Java Servlet Container, e.g. Apache *Tomcat* (deploy)
- dose-response-jni

Initial installation

Download the project source and go to this component's root directory (i.e. [here](#)) and follow the steps below.

1. [T] Copy dose-response-jni target/dose-response-fitter.jar to lib/dose-response-fitter.jar
2. [T] Copy dose-response-jni target/libfittingdoseresponse.so to lib/libfittingdoseresponse.so
3. [T] Copy src/properties/sample.env.properties to src/properties/env.properties

Configuration

src/properties/env.properties

```
soap.location=
```

WS URL of this component, e.g. 'http://localhost:18180/fdr_manager-0.0.1-SNAPSHOT/fwsw/'

```
log.file=
```

Component log file location on disk, e.g. 'logs/dose-response-manager.log'.

```
log.level=
```

Component log level, e.g. (trace|debug|info|warn|error|fatal).

Build

-P it

The *Maven* profile to use. At the moment the pom.xml defines only the use of the it profile for integration testing.

Example build instructions :

```
cd <ap_predict_online>/dose-response-manager
mvn -P it clean verify
```

Note: Building with the integration testing activated will require that the “.so” files of libfittingdoseresponse.so and libjson can be found in the system library path. e.g export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:<ap_predict_online>/dose-response-manager/lib:/home/me/mylibjson/lib

Deploy

If the build command has completed successfully the webapp .war file should be in the `target` directory. Copy this to the servlet container's relevant file, e.g for *Tomcat*, the `webapps` directory.

Note: As in the *Build* situation, the “.so” files of `libfittingdoseresponse.so` and `libjson` will need to be found by the deployed .war file. For *Tomcat* this would mean something like the following before start-up. #. Copy `libfittingdoseresponse.so` to a “lib” directory #. `export JAVA_OPTS="-Djava.library.path=lib"` (i.e. the directory in the previous step). #. `export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/me/mylibjson/lib`

Run

See *start*.

Start-up problems

`java.lang.UnsatisfiedLinkError: no fittingdoseresponse in java.library.path`

Indicates that `libfittingdoseresponse.so` can't be found in the `java.library.path` (which is generally used by Java to locate native libraries). Ensure that the appropriate `JAVA_OPTS="-Djava.library.path=<directory of libfittingdoseresponse.so>"` value is set before starting.

site-business

`site-business` is the final phase of building the business logic after building `business-manager-api` and then `business-manager`. The `site-business` component in the `ap_predict_online` repository represents a compound database and associated site data processing **of a fictional company**. If you wish to build your own `site-business` see *here*.

Dependencies

- *Maven* (build)
- Java 7 or higher (build, deploy)
- Java Servlet Container, e.g. Apache *Tomcat* (deploy)
- Database (build, deploy)
- `business-manager-api` and `business-manager`

Database choice

See also:

Database choice

Initial installation

Download the project source and go to this component's root directory (i.e. [here](#)) and follow the steps below.

1. [T] Copy `src/properties/sample.env.properties` to `src/properties/env.properties`
2. [T] Copy `src/properties/database/site/sample.site.database.properties` to `src/properties/database/site/site.database.properties`

Configuration

`src/properties/env.properties`

```
log.file.site_business=
```

Component log file location on disk, e.g. 'logs/site-business.log'.

```
log.level.site_business=
```

Component log level, e.g. (trace|debug|info|warn|error|fatal).

```
log.level.general=
```

Log level of code from other libraries, e.g. (trace|debug|info|warn|error|fatal).

`src/properties/database/site/site.database.properties`

If you wish to use the embedded *HSQL* database then this file can have all content removed, otherwise you will need to assign the appropriate JDBC driver and user details.

Build

`-Dspring.profiles.active=`

Options are currently (sitedata_embedded|sitedata_mysql).

Warning: At build time it is **important** to use the `-Dspring.profiles.active=sitedata_embedded` arg because during integration testing rubbish may be written to the database so it is important to use the embedded database during this process.

Example build instructions :

```
cd <ap_predict_online>/site-business
mvn clean verify -Dspring.profiles.active=sitedata_embedded
```

Deploy

If the build command has completed successfully the webapp .war file should be in the target directory. Copy this to the servlet container's relevant file, e.g for *Tomcat*, the webapps directory.

Run

See *start*.

Start-up problems

TODO

4.1 Security

4.1.1 Communication security

Note: Communication security is only really essential for the communication to `site-business` as that component should only process requests containing compound identifiers arriving from the `client` component (or any other suitably secured “client”, e.g. a Perl script) which is *expected* to have an appropriate authentication mechanism configured. See [here](#) for further information on generic client authentication options. For `app-manager` and `dose-response-manager` component WS's, these could be kept as “open” resources for any potential “client” application, e.g. a command-line script, as the WS message content will never contain any data which identifies a compound.

HTTPS

All inter-component WS communication uses [WSS](#) and so all such communication should be encrypted HTTP (Hypertext Transfer Protocol), e.g. using X.509 certificates (self-signed or otherwise).

An example configuration of such a setup is available for [Tomcat](#).

WSS

All inter-component WS communication uses WSS ([description here](#)) of the `User ID/Password` variety. For this to be effective the passwords used should be :

1. *Stored in an encrypted format*
2. Communicated in an encrypted format, i.e. [HTTPS](#).
3. *In access-controlled files*

WS IP (Internet Protocol) access control

Access to any component's WS, e.g. `app-manager`, `site-business` and `dose-response-manager`, can be restricted to specified IP origin addresses by, for example, the inclusion of a [Remote Address Filter](#) in *Tomcat*'s `web.xml` configuration file.

WWW/Internet communication

Once installed, AP-Portal's *generic portal components* have no need to access the internet – the portal could operate unhindered within corporate firewalls. Having said that, `ApPredict` on the other hand will attempt to retrieve *variability lookup tables* if they cannot be found locally (see *variability configuration*).

Having said that, there would be nothing preventing *site-specific* component implementations or extensions, e.g. for `client` or `site-business`, from communicating through corporate firewalls if requirements necessitate doing so.

4.1.2 Database security

Access to whichever database which AP-Portal is configured to use should be appropriately controlled to ensure that the content can not be accessed. This is particularly the case for the `site-business` component which stores compound identifiers alongside simulation input values and results, but equally for `client-direct` which retains the portal usernames and passwords.

This risk is avoided if there is suitable *Filesystem access control*.

4.1.3 AP-Portal dependency security

The portal, like any software, does have a number of library dependencies which may contain security vulnerabilities which are presently unknown or may be disclosed in the future.

The portal developers will attempt to keep up-to-date with emerging potential security vulnerabilities, e.g. via <https://nvd.nist.gov/download/nvd-rss-analyzed.xml>, but even if a vulnerability appears it may not be possible to update the portal software in a reasonable time frame.

Dependencies are listed in the various component Maven `site` build reports no longer available except my building the components.

Useful tools

If you wish to determine the most up-to-date security statuses for any of the dependency libraries of components then there is the following which may be useful :

```
mvn org.sonatype.ossindex.maven:ossindex-maven-plugin:audit -f pom.xml
```

4.1.4 Filesystem security

Filesystem access control

Security is improved by ensuring that files which contain sensitive information or data are appropriately access-controlled at a filesystem level, for example, use of the `chmod 600 <file_name>` command.

Warning: It would be easier to create a specific account for `AP-Portal` and restrict access to that account's directory solely to that account (rather than, for example, also allowing group or unrestricted access).

JASYPT file content encryption

See *Property file {de,en}encryption*.

This risk is avoided if there is suitable *Filesystem access control*.

Log files

With log levels of `debug` or `trace` the log files may contain a large amount of data, some of which may be sensitive so it is advised that under normal operation a log level of at least `info` is used.

This risk is avoided if there is suitable *Filesystem access control*.

4.1.5 GUI (Graphical User Interface) security

The `client` and `client-direct` component both undertake user input and output validation and “sanitisation”.

5.1 Personal Data

No aspect of `AP-Portal` operation involves the storage or use of PII (Personally Identifiable Information) nor any form of personal data.

6.1 Configuration

Note: If you're reading this because a part of the portal is not working then please also consider visiting the [troubleshooting](#) pages.

Note: Generally changes to `ApPredict` are reflected in `AP-Portal`, which is good news for anyone trying to keep up-to-date with the latest developments, however there is currently no mechanism of version-checking between, for example, the `app-manager` component and the version of `ApPredict` which it invokes. It is possible that the `ApPredict` could undergo many changes without the need to update `AP-Portal`, and vice-versa. Similarly, there are no checks in place prior to deployment which verify that communication between the `AP-Portal`'s *generic portal components* is compatible (although generally, any incompatibilities that existed would soon appear at runtime as system errors!).

Initial component configuration is handled in the component's *installation* section¹.

6.1.1 Specific Functionality

Variability Configuration

See also:

¹ `AP-Portal` development up to now has predominantly focused on the ability to quickly adapt to frequently changing requirements to develop this application, with developers being actively involved in all aspects of `AP-Portal` installation, support and maintenance. Whilst this approach has been effective in terms of being able to quickly deliver and advance a proof-of-concept application, many aspects of configuration take place at a systems level at install or upgrade time, e.g. XML file content is modified before components are (re)built and (re)deployed. Once configured the `AP-Portal` provides the users with an environment which meets their requirements (although there is a growing list of feature requests), however a number of what would be considered infrequent "business" configuration changes, e.g. a different CellML model is to be assigned as the default, or a different units assigned as the default, would require the modification at a filesystem level and a component restart. In due course the majority of anticipatable configuration changes will be available via the UI.

Related research paper: [Journal of Pharmacological and Toxicological Methods](#)

In order for variability data to appear in simulation results in the UIs of both the `client` and `client-direct` components, ApPredict needs to have made available to it *variability lookup tables* - a manifest of which is available from [Gary Mirams' repository](#).

Currently (as of Jan. 2018), whenever ApPredict is run with the `--credible-intervals` (which itself had its functionality extended May 2018 to allow the specification of credible interval percentile values) and `--model <model id>` args supplied, it will look in ApPredict's current working directory¹ for an unpacked version² of the relevant lookup table. If it is not found ApPredict will attempt to download the packed file from the aforementioned repository and if such a file exists it will be downloaded, unpacked locally, and loaded into memory³.

¹ This has been achieved by modifying `app-manager's prepare.sh` file to symlink to manually downloaded unpacked *variability lookup tables* in ApPredict's current working directory. Ideally it is the `*_BINARY.arch` files that should be made available (as these are created dynamically from the non-BINARY `.arch` files when ApPredict loads them), although there may be compatibility issues using legacy versions derived from historical compilations.

² Conventionally the *variability lookup tables* are released in `.tgz` format due to their large (e.g. 700Mb+) size.

³ This has the potential to drain the hardware's available RAM which may impact server performance.

7.1 Maintenance

7.1.1 General

Component Start

Example start script

```
#!/bin/bash -e

if [ $# -ne 1 ]; then
    echo ""
    echo "  Usage: start.sh <component>"
    echo "    e.g. start.sh app-manager"
    echo ""

    exit 1
fi

component=$1

if [ -d ${component} ] ; then
    rm -rf ${component}/{logs,temp,work}/*
    rm -f logs/${component}.log*
fi

. `pwd`/${component}.env
`pwd`/bin/catalina.sh start
```

See also:

Environment setting below (for adjusting, for example, CATALINA_OPTS).

Component Stop

Example stop script

```
#!/bin/bash -e

if [ $# -ne 1 ]; then
    echo ""
    echo "  Usage: stop.sh <component>"
    echo "        e.g. stop.sh app-manager"
    echo ""

    exit 1
fi

component=$1

. `pwd`/${component}.env
`pwd`/bin/shutdown.sh
```

Component Restart

See *Component Start* and *Component Stop*!

Environment setting

Default values for Java memory allocations may be available using, for example, `java -XX:+PrintFlagsFinal -version | grep 'PermSize'`.

In the example scripts above reference is made to a *local* environment-setting file. An example of such file, e.g. `app-manager.env`, for *Tomcat*¹ use follows :

```
JAVA_HOME=/usr/java/current
CATALINA_HOME=/home/me/apps/tomcat/CATALINA_HOME
CATALINA_BASE=/home/me/apps/tomcat/vhosts/app-manager

# Xmx/Xms = JVM heap (max/initial). Object instance storage. Equal values == "fully_
↪committed" - avoids GC as initial expands to max.
# Xss      = Thread stack size. If you get a StackOverflow exception, increase this_
↪value!
# -XX:PermSize -XX:MaxPermSize = Class files storage.
#CATALINA_OPTS="-Xms256m -Xmx256m -Xss256k -XX:PermSize=64m -XX:MaxPermSize=64m"

# Set in a 32-bit Win environment. Auto-enabled on 64-bit OSs
# JAVA_OPTS="-server"
JAVA_OPTS="-Dspring.profiles.active=app_manager_mysql"

JASYPT_PWD=<password>

export JAVA_HOME JAVA_OPTS CATALINA_BASE CATALINA_HOME JASYPT_PWD
```

See also:

¹ In earlier versions of AP-Portal the CATALINA_OPTS option was trialled to test the minimum memory required but since the introduction of PK processing (around late 2017) the memory requirements of most components increased significantly and so memory requirements may need to be specified towards the higher ranges for some components (`client` for example may require `-XX:MaxPermSize:512m`).

Java troubleshooting.

7.1.2 Updating – Remote AP-Portal/ApPredict changes

These changes reflect changes in the source code of the generic functionality.

See also:

An *example updating strategy*.

General updating concepts

The updating process will involve some or all of the steps below and will *at least* involve the portal's technical administrator and potentially a scientist responsible for the portal's scientific settings (e.g. someone who would know information such as the assays and ion channels being processed) :

The steps are :

1. Reflect any new files in the *portal repository* source code onto the *local* system, i.e., create files according to any modified installation instruction documentation file(s).
2. Reflect any modifications to the *portal repository* `sample.<file_name>` files onto the *local* configuration files (which were originally created during *installation*). Details of files which may have been modified may be found in the *ChangeLog*.
3. Reflect any modifications to the *portal repository* files onto *site-specific* (including bespoke) code and configurations.

This may be necessary if, for example, the `business-manager-api` or `business-manager` code changes, and there is bespoke, *site-specific* code which references any of the changed objects or settings.

4. If you have more than one installation environment to maintain, e.g. *development*, *test* and *production*, then for each update in the feeder environment please record the modifications made so that accumulated changes can be applied to the consumer environment.

In theory, a scientist would only need to present for the update of a *development* environment as changes to the *test* and/or *production* environment could be derived from recorded decisions made during the update of the *development* environment.

See *here* as an example.

Retrieving and analysing changes in portal source code

In order to update the *generic portal components* the standard procedure is to execute the following `git` commands in sequence at the base of the cloned AP-Portal directory. A collection of the latest command examples is stored in the *tools section* :

1. `git fetch`

(Outcome: Does not update the *local* source code files or directories , only updates git's internal version control system, but allows us to anticipate what changes will be required locally.)

2. `git diff --name-status HEAD...<git hash>, or git diff --name-status HEAD... origin`

(Outcome: Lists the source code file changes and the nature of change)

Generally we focus on the (M)odified, (D)eleted or (A)dded of the following file types, but also check the *ChangeLog*:

1. sample.<file_name>
2. .gitignore
3. <file_name>.xsd
4. installation/components/<component>/index.rst

A sample command would be `git diff --name-status HEAD...origin | grep -P '(/sample\.|\.gitignore|\.xsd|installation/components/.*/index\.rst$|/appCtx\.config\.|/pom\.xml|\.sql) '`

If any changes have been made to the source code of such files please record their names for closer examination in the next step.

3. `git diff HEAD...<git hash> <file_name>`, or `git diff HEAD...origin <file_name>`

(Outcome: Prints a diff output of the changes of a specified file if it has been (M)odified since the last git merge origin.)

Determine what, if any, *local* changes will be required **after** an instruction to “merge” the update (see next step) is executed. See *Updating configuration and settings files* and *Updating web services*.

4. `git merge <git hash>`, or `git merge origin`

(Outcome: Updates the source code files and directories.)

Updating configuration and settings files

During the original installation of each of the components it is very likely that *site-specific* configurations were derived from generic configuration or settings files, e.g. files such as `src/properties/filter.properties` derived from their corresponding “sample” file `src/properties/sample.filter.properties`, or in the case of the client portal, *Bespoke site-business functionality*. Invariably these instructions were defined in the component’s installation instruction file.

If the content of the “sample” files has been changed by the update request (e.g. perhaps a new configuration option has been added to a component) then any such changes will need to be implemented in the derived *site-specific* implementations of those files.

At this stage the best option is to view the output of the `git diff --name-status HEAD...origin` command to check which of the files listed in a component’s installation instructions documentation file have been modified and manually apply changes to the *local* version.

Updating web services

If any of the components’ WS XSD (XML Schema Definition) files have been modified (usually located in the component’s `src/main/resources/META-INF/schema/` directory) then, if the change results in a change of contract (as opposed to a change of documentation) of the component’s WS API, any components with which it communicates *may* also need to be updated.

Restarting components

Once any changes have been made to configuration and settings files the components need to be rebuilt (see Build instructions in *Installation*), redeployed and *restarted*.

app-manager updating

First take a look at the *general update instructions*.

Based on what has appeared previously in app-manager's *installation instructions*, *local* versions of the following files are likely to require updating if the corresponding sample. `<file_name>` files have changed in the *portal repository*.

1. `src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-incoming.xml`
2. `src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml`
3. `src/properties/database/database.filter.properties`
4. Files derived from `src/properties/database/sample.database.spring.properties`
5. `src/properties/filter.properties`
6. `src/properties/spring.properties`
7. `pom.xml`

Warning: Both client-direct and/or business-manager communicate with the app-manager WS API. If app-manager's WS XSD file (`app_manager.xsd`) has changed, and the change results in a new WSDL (Web Service Definition Language) contract, then it will require client-direct and/or business-manager (and hence site-business) to be rebuilt.

See also:

If you are updating app-manager then it is possible that you may also need to *update ApPredict*

business-manager updating

First take a look at the *general update instructions*.

Based on what has appeared previously in business-manager's *installation instructions*, *local* versions of the following files are likely to require updating if the corresponding sample. `<file_name>` files have changed in the *portal repository*.

1. `src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-incoming.xml`
2. `src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml`
3. `src/properties/database/database.filter.properties`
4. Files derived from `src/properties/database/sample.database.spring.properties`
5. `src/properties/filter.properties`
6. `src/properties/spring.properties`
7. `pom.xml`

Warning: Both client and/or app-manager communicate with the business-manager WS API. If business-manager's WS XSD file (`business_manager.xsd`) has changed, and the change results in a new WSDL contract, then it will require client and/or app-manager to be rebuilt.

business-manager-api updating

First take a look at the *general update instructions*.

Generally any modifications to files in this component will result in possible changes necessary only in dependent components, i.e. business-manager or site-business.

If there has been a change in business-manager-api then downstream components (e.g. business-manager and/or site-business) will also need updating.

client updating

First take a look at the *general update instructions*.

Based on what has appeared previously in client's *installation instructions*, *local* versions of the following files are likely to require updating if the corresponding sample.<file_name> files have changed in the *portal repository*.

1. src/main/resources/bundle/site.properties
2. src/main/resources/META-INF/spring/ctx/config/appCtx.config.site.xml
3. src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml
4. src/properties/database/database.filter.properties
5. Files derived from src/properties/database/sample.database.spring.properties
6. src/properties/filter.properties
7. src/properties/spring.properties
8. pom.xml

If you have a client-specific “prepopulated user authentication” *user authentication mechanism* database then also check the following :

1. src/main/resources/META-INF/data/spring-security/local/users.sql

client-direct updating

First take a look at the *general update instructions*.

Based on what has appeared previously in client-direct's *installation instructions*, *local* versions of the following files are likely to require updating if the corresponding sample.<file_name> files have changed in the *portal repository*.

1. src/main/resources/META-INF/spring/ctx/config/appCtx.config.cellModels.site.xml
2. src/main/resources/META-INF/spring/ctx/config/appCtx.config.site.xml
3. src/main/resources/META-INF/spring/ctx/ws/appCtx.ws.security-outgoing.xml
4. src/main/webapp/resources/js/site/site.js
5. src/properties/database/database.filter.properties
6. Files derived from src/properties/database/sample.database.spring.properties
7. src/properties/filter.properties
8. src/properties/spring.properties
9. pom.xml

If you have a `client-direct-specific` “*prepopulated user authentication*” *user authentication mechanism* database then also check the following :

1. `src/main/resources/META-INF/data/spring-security/local/users.sql`

client-parent updating

First take a look at the *general update instructions*.

Generally there shouldn't need to be any manual changes to this component as part of an update.

client-shared updating

First take a look at the *general update instructions*.

Based on what has appeared previously in `client-shared`'s *installation instructions*, *local* versions of the following files are likely to require updating if the corresponding sample `<file_name>` files have changed in the *portal repository*.

1. `src/main/resources/META-INF/data/spring-security/local/users.sql`
2. `src/main/resources/META-INF/resources/resources/css/site/client-shared-site.css`
3. `src/main/resources/META-INF/resources/WEB-INF/spring/ctx/config/appCtx.features.xml`
4. `src/main/resources/META-INF/resources/WEB-INF/tiles/layout/common/logo.jsp`
5. `src/main/resources/META-INF/resources/WEB-INF/tiles/layout/common/logout.jsp`
6. `src/main/resources/META-INF/resources/WEB-INF/tiles/layout/contact/contact.jsp`
7. `src/main/resources/WEB-INF/spring/authn/appCtx.bespoke.xml`
8. `src/main/resources/WEB-INF/spring/authn/appCtx.sitePreauthFilter.xml`
9. `src/main/resources/WEB-INF/spring/root-context.site.xml`
10. `src/properties/cs-filter.properties`
11. `pom.xml`

dose-response-jni updating

First take a look at the *general update instructions*.

Based on what has appeared previously in `dose-response-jni`'s *installation instructions*, *local* versions of the following files are likely to require updating if the corresponding sample `<file_name>` files have changed in the *portal repository*.

1. `makefile`

If there has been a change then downstream components (e.g. `dose-response-manager`) will also need updating.

dose-response-manager updating

First take a look at the *general update instructions*.

Based on what has appeared previously in dose-response-manager's *installation instructions*, *local* versions of the following files are likely to require updating if the corresponding sample .<file_name> files have changed in the *portal repository*.

1. env.properties

Warning: business-manager communicates with the dose-response-manager WS API. If dose-response-manager's WS XSD file (*fdr_manager.xsd*) has changed, and the change results in a new WSDL contract, then it will require business-manager (and hence site-business) to be rebuilt.

site-business updating

Note: These instructions refer to the site-business of the *generic portal components*, and are unlikely to require updating unless you're demo'ing the fictional company compounds. If you're looking for instructions for a *site-specific* site-business update then please visit the installation information contained in *AP-Portal extensibility*.

First take a look at the *general update instructions*.

Based on what has appeared previously in site-business's *installation instructions*, *local* versions of the following files are likely to require updating if the corresponding sample .<file_name> files have changed in the *portal repository*.

1. env.properties
2. src/properties/database/site/site.database.properties

ApPredict updating

If there's a new version of ApPredict then there are a number of issues to consider which may have a greater impact, such as :

- If we were to update to the latest version of ApPredict to allow us to use the new CellML models, has the ApPredict developer also made other, potentially significant, changes which will require a new version of AP-Portal to be installed?
 1. If ApPredict has changed considerably there's most likely a new version of AP-Portal (or at least app-manager to install - see *update app-manager*).
 2. If the ApPredict invocation mechanism has changed then perhaps it will be necessary to update the *app-manager tools* e.g. CATALINA_HOME/ApPredict.sh
- If ApPredict has added a new CellML model to the available collection (i.e. has ApPredict's --model range expanded), are there also new *variability lookup tables* available for download?
- Does the new CellML model behave differently to existing CellML models? If it does (i.e. perhaps it is adversely affects the performance of AP-Portal), then please let the portal developers know¹!

¹ Any suggestions for change or queries would be welcomed by the portal developers!

ApPredict has a new CellML model available ...

In the scenario that the only change to ApPredict is a new CellML model (or models) which behave much like existing models, and new *variability lookup tables* are available, please complete the following :

1. Install the new ApPredict (making sure not to provisionally avoid overwriting/removing the operational version) - see *ApPredict installation*.
2. Download any new *variability lookup tables* and deploy them as per previous lookup table installation - see *Variability Configuration*.
3. Whichever portal (client and/or client-direct) you are using, CellML model data is contained in the `src/main/resources/META-INF/spring/ctx/config/appCtx.config.cellModels.site.xml` file :
 - a. client portal Adjust the `site-business` file as per the *extensibility instructions* to include the new CellML model details.
 - b. client-direct portal Adjust the `client-direct` file as per the *installation instructions* to include the new CellML model details.

ChangeLog

Changes in the main configuration files.

Unreleased

Date		Component	Change	Git Hash - View diff

7.1.3 Updating – Local changes

These changes generally reflect changes in user preferences or changes to *site-specific* configurations, e.g. new database connection settings.

See also:

An *example updating strategy*.

site-business

Changing database connectivity

There are two ways to change the database connectivity, both of which can make use of the *property file encryption* if the data is sensitive :

1. Temporary change, i.e. the change will be overwritten on a subsequent redeployment unless the permanent change is undertaken.

If `site-business` has been deployed in the servlet container (e.g. the `site_business-<version>.war` file has been expanded in the `webapps` directory), then modify the content of the file `site_business.properties`, e.g. `CATALINA_HOME/site-business/webapps/site_business-<version>/WEB-INF/classes/META-INF/properties/site_business.properties` and *restart* `site-business`, upon which `site-business` will attempt to connect to the new database.

2. Permanent change.

For a change to become effective when site-business is *redeployed*, the site-business source code file `<ap_predict_online>/site-business/src/properties/database/site/site.database.properties` needs to be assigned the desired values, then the source code needs to be *rebuilt* and *redeployed*.

7.1.4 Example User Requests

See also:

An *example updating strategy*.

Example client-only Portal User Requests

Please include a new assay in the results.

If a new assay is to be introduced then the scientists will need to determine a number of aspects prior to making any change in the portal configuration or code, e.g. :

1. What is the definitive name of the assay?
2. Where in the *assay level* hierarchy of assays is this new assay to appear?
3. Which *assay group* is the assay to belong to?
4. How will the assay data be retrieved?
5. Will the data associated with this assay be the same as other, existing assays, or will it differ in, for example, units, representations, processing requirements, *individual data* status, etc.? If the assay will be generating results in an identical manner to other assays then there won't be a need for much additional code to be written.

Essential

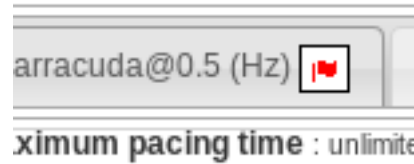
1. `appCtx.config.assays.site.xml`. See the *extensibility instructions* related to assay details.
2. Create additional code reflecting the technique, e.g. SQL, WS, for reading in the new assay data and, if necessary, additional processing before passing such data over to the business-manager (via `business-manager-api` structures).

Optional

1. Create new `datarecord` implementations if necessary. See the *extensibility instructions* related to defining *site-specific* data record structures derived from `business-manager-api`.
2. If there is a significant change of processing required for this new assay then processing steps such as those demonstrated in the generic *site-business* may be required, e.g. as in 4 - *Define Screening, QSAR, Experimental processing*.

Warning: If a new assay is to be included then the simulation database needs to be purged of all existing simulation results because previously used assay identifiers which are persisted in the simulation results will (more than likely) no longer be valid.

The simulation's failed to run - why?



There is a hidden option to *right-click* whilst over the job flag, i.e. [Source](#), after clicking (or re-clicking) the *submit* button. This will expand the diagnostics details at the base of the page, which can be removed either by repeating the *right-click* or by choosing a different simulation.

If you've arrived at this page directly it may be worth briefly taking in some of the general [configuration](#) notes.

Similarly, if you're receiving requests which are likely to require updating AP-Portal to the latest public version additional, general notes are available in the [Updating – Remote AP-Portal/ApPredict changes](#) section of this documentation.

Please change the values of ion channel variability.

- `client` The [variability](#) values are defined in the `site-business appCtx.config.assays.site.xml` file as per the [extensibility instructions](#).
- `client-direct` The visible default values of spread [variability](#) are defined in the `client-direct appCtx.config.site.xml` file as per the [installation instructions](#).

Note: See³ regarding the change deployment technique, and⁴ for additional information.

Please change the default CellML model.

This is set by having only one of the available models assigned a `defaultModel` value of `true` in the following :

- `client` The `site-business appCtx.config.cellModels.site.xml` file as per the [extensibility instructions](#).
- `client-direct` The `client-direct appCtx.config.cellModels.site.xml` file as per the [installation instructions](#).

Note: See³ regarding the change deployment technique.

Please change the values of credible interval percentiles.

The [variability](#) percentile values are defined in the following :

- `client` The `site-business appCtx.config.actionPotential.site.xml` file as per the [extensibility instructions](#).

³ Modification of the configuration files should ideally be done prior to rebuilding and redeploying the relevant component (as opposed to directly modifying the content of an expanded `.war` file in the servlet container and restarting the component).

⁴ See also [Variability Configuration](#).

- `client-direct` The `client-direct` `appCtx.config.site.xml` file as per the *installation instructions*.

Note: See³ regarding the change deployment technique, and⁴ for additional information.

Please modify information displayed by the information icons.



Currently the information displayed by  is not configurable.²

Information icons appear in various places throughout the `client` and `client-direct` UIs. Modification of the information they display is generally not possible as they are designed to enable the provision of an *internationalised* AP-Portal. It is of course possible to directly change the AP-Portal code base, rebuild and redeploy, but if tempted to do so please be careful regarding *version control conflicts*). Having said that, parts of the values they display may be flexible, e.g. the sample below is taken from a `middle.jsp` file in `client-direct`.

```

1 <td colspan="4" class="input_division cs_rounded_5">
2   <spring:message code="general.notes" />
3   " /> />
6 </td>

```

In the above we can see the use of the `<spring:message code="general.notes" />` which, when the UI is to display the corresponding web page, will read in and display the `general.notes` property value derived from the `src/main/resources/bundle/general[_locale].properties` files¹. What is also visible is the use of the `<spring:message code=".." arguments="<args>" />` option which allows dynamic replacement of values which may be derived from, for example, values defined in configuration settings and so could be a technique for future (non-I18N!) information display.

Please add the new CellML models the ApPredict developer has released.

Please see *ApPredict updating*.

Please modify the warning/error messages.

As is the case with *Please modify information displayed by the information icons*, the warning messages are general not modifiable except by request to the AP-Portal developers. Whilst some error messages are *internationalised* some are also derived from hard-coded messages in *back-end components*.

Please update AP-Portal to incorporate the new development in the public version.

There are unfortunately no *one-size-fits-all* step-by-step guides to doing this unless it's known what the modifications to the AP-Portal have been between the currently installed version and the latest available public version.

Some *general updating notes* are available which may provide guidance.

² Any suggestions for change or queries would be welcomed by the portal developers!

¹ The "locale" is considered to be the browser's current preferred display language.

7.1.5 Example IT Dept. Requests

See also:

An *example updating strategy*.

If you’ve arrived at this page directly it may be worth briefly taking in some of the general *configuration* notes.

We need to restart AP-Portal’s own databases, do we need to shut down the components?

The short answer is ‘yes’.

The reason being that of the *generic portal components*, `app-manager` and `business-manager` frequently poll the database to find newly persisted data (e.g. a new simulation request). In doing so a momentarily disappeared database may cause some issues (albeit probably not fatal ones) if the portal is running , even if there are no users using the portal at the time!

7.1.6 Example strategy for updating many hosted environments

Preamble

The policy adopted in this example strategy is that there are two mechanisms for managing *site-specific* data, these are :

1. Environment-agnostic code, configurations and settings which do not contain sensitive data (e.g. passwords) are placed in a *subversion* repository accessible to all servers. The assumption here being that subversion repository is a “public” facility which is not sufficiently secured to guarantee that portal code will be accessible only to AP-Portal developers/administrators.
2. Environment-specific code, configurations and settings, and objects containing sensitive data, are placed in the component sub-directories of (the *git*-managed) `ap_portal_online`.

When building for a particular environment components which are ...

1. *generic portal components*: That component’s subversion-resident objects are overlayed over its corresponding *git*-managed `ap_portal_online` sub-directory code just prior to invoking the build instruction in the cloned git directory.
2. *site-specific*: That component’s building takes place in the subversion repository code working directory.

Hardware

The example hardware configuration is as follows :

Server	Environment
server1	development
server1	test
server2	production

Version control system

A *subversion* repository, accessible to all servers , hosts the *site-specific* code, configurations and settings which **don’t** contain sensitive data and are environment-agnostic, i.e. objects created in the *development* environment which can be deployed without modification during updates to *test* and *production* environments.

For example, the structure of the subversion repository would be similar to the following, whereby the *<git hash>* would be the ap_predict_online commit short hash used at the time of building the update :

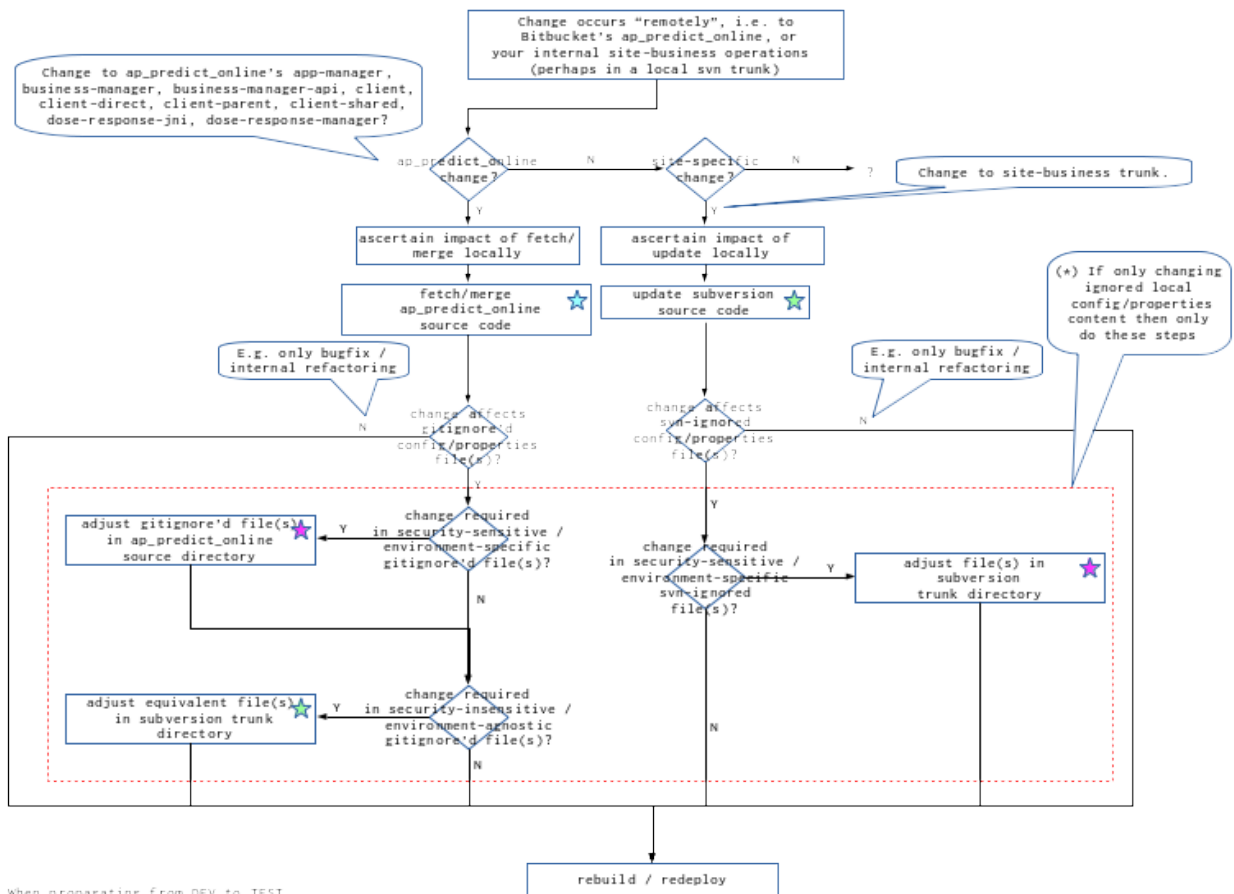
```

svnrepo/client/branches
|      +---/tags/development      --+
|      |      |      +-/160912_<git hash 1>      |
|      |      |      +-/160930_<git hash 2>      |
|      |      |      +-/161015_<git hash 3>      |
|      |      |      +-/161016_<git hash 4>      |
|      |      |      +-/161030_<git hash 5>      |
|      |      +---/production      |-- Records of historical tagged_
↔updates
|      |      |      +-/160925_<git hash 1>      |
|      |      |      +-/161022_<git hash 4>      |
|      |      +---/test      |
|      |      |      +-/160920_<git hash 1>      |
|      |      |      +-/161002_<git hash 2>      |
|      |      |      +-/161020_<git hash 4>      --+
|      |
|      +---/trunk/pom.xml      --+
|      |      +---/src/main/java      |
|      |      |      |      +-/com      |
|      |      |      |      +-/resources      +--- Latest development code
|      |      |      |      +-/webapp      |
|      |      |      +-/test/java      |
|      |      |      +-/com      --+
|
|      +---/site-business/branches      --+
|      |      +-----/tags      +--- Bespoke site-business_
↔component
|      |      +-----/trunk      --+

```

Update cycle

Updating DEV source code (or config/properties values *)



When propagating from DEV to TEST....

- ★ Changes need to be manually replicated on TEST.
- ★ Changes need to be subversion tagged and those tags referenced by build and reset scripts on TEST.
- ★ Equivalent fetch/merge(s) need to be made on TEST.

Fig. 1: Sample dev -> test update considerations.

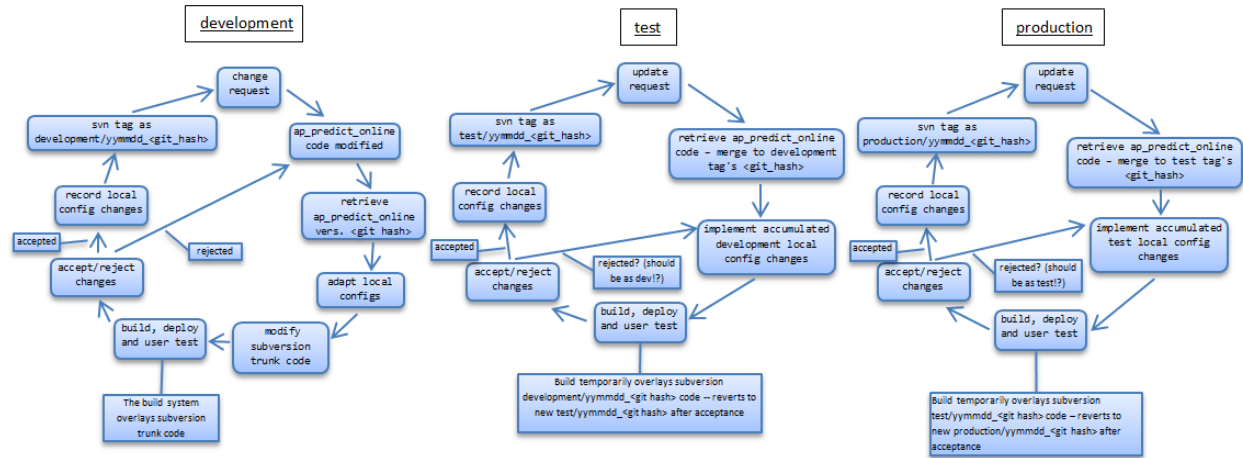


Fig. 2: Sample update cycle.

8.1 General

8.1.1 Version Control

Conflicts due to local file changes

If version-controlled files or directory structures are changed locally, e.g. file content changed or new files/directories added, then this potentially causes conflicts if `AP-Portal` is updated to the latest release of the software.

9.1 Tools

9.1.1 HSQL

HSQL's website is [here](#)

HyperSQL is a 100% Java database which, in this application, is generally used as an embedded (i.e. you don't need to set up database usernames and passwords, the db is only in existence for as long as the application is running) database either just during build processes, or for some components (heads-up!), in a deployment environment.

9.1.2 Maven

Maven's website is [here](#), it's Java-based and is used to build projects from source.

Note: The instructions below are for a version of MySQL which by the time you read this may have security issues!

9.1.3 MySQL

MySQL's website is [here](#)

Example install

Note: Other databases can be used as AP-Portal components use [Hibernate](#), but alternatives haven't been tested.

1. Estimated time to build : 2 minutes.
2. Desired structure : Similar (eventually) to the following :

```
<data_dir>
|
+-----/app_manager
+-----/business_manager
+-----/client_direct
+-----/etc/init.d/mysql_apportal
+-----/innodb
+-----/my.cnf
+-----/mysql
+-----/var
    +--/lib/mysql/mysql.sock (when running)
    +--/log/mysql/mysql-bin.
    |      +--/mysqld-log
    |      +--/mysqld-log.err
    +--/run/mysqld/mysqld.pid (when running)
```

1. `cd <any empty temporary directory>`
2. Download [MySQL install scripts](#), retaining directory structure.
3. Run `linstall.sh <mysql binary location> <database dir> <system_user> <system_group>` e.g. `linstall.sh /apps/mysql/5.6.27 /data/apportal user_me group_me`

9.1.4 Oracle

Oracle's website is [here](#)

To emulate Oracle use it is possible to register for and download Oracle XE (*Express Edition*) which can be package-installed, for example, into a VM (Virtual Machine) and networked to appear as a local database.¹

If you're downloading Oracle stuff, you'll probably also need to download the appropriate JDBC (Java DataBase Connectivity) driver and more than likely the *SQL Developer* application to view table structure and content.

9.1.5 Property file {de,en}ryption

1. Prior to early 2019 the `client` and `client-direct` components were able to use *Jasypt* encryption but since that time some dependency library updating for security purposes has meant that with *Spring* v.5 it was no longer possible (or rather, with there being no *Jasypt* activity since 2014 it was decided not to continue using it).
2. Download and unpack <http://www.jasypt.org/download.html> (current version is 1.9.2) and run the relevant `bin` directory `.bat` or `.sh` scripts. (AP-Portal uses the `org.jasypt.encryption.pbe.StandardPBEStrngEncryptor` class, as do the scripts).

```
encrypt.sh input=<property file value> password=<jasypt encrypting pwd>
```

e.g. `./encrypt.sh input=fish password=chips.`

Note: You may prefer the above command containing your password not to appear in your shell history, in which case various techniques are available, e.g. check [stackoverflow](#).

3. The output generated then needs to be placed into the relevant `.properties` file, e.g.

Sample `encrypt.sh` output...

¹ I used *Oracle Database Express Edition 11g Release 2* installed in a *VirtualBox* Fedora 20 for testing purposes - although I did need to [increase swap space](#) and tried `alter system set processes=200 scope=spfile` when in the `sqlplus` system CLI (Command-line Interface).

```

----ARGUMENTS-----

input: fish
password: chips

----OUTPUT-----

MHWlsEFq4rI/Rx7s1H27pg==

```

... is placed into a `spring.properties`:

```
securement.business.password=ENC(MHWlsEFq4rI/Rx7s1H27pg==)
```

4. `<jasypt encrypting pwd>` needs to be assigned to the environment variable `JASYPT_PWD` prior to the *component* being started, e.g. a component's start file may look something like:

```

export JASYPT_PWD=chips
./bin/catalina.sh start

```

9.1.6 Spring

Spring profiles

Spring profiles are used as an easy way to have different application configuration options active in different environments without a need to modify the codebase or recompile, instead the configuration to be used is determined by the passing of the `spring.profiles.active` Java system property to the application (e.g. by use of the `-Dspring.profiles.active=<profile1>,<profile2>,<etc>` arg when starting the portal from the command-line). Whilst this means that all anticipated configurations are packaged it does permit a degree of flexibility so that, for example, a single package can be used in both the build and deployment environment.

Note: The instructions below are for a version of Tomcat which by the time you read this may have security issues!

9.1.7 Tomcat

Tomcat's website is [here](#)

Example install

Note: Other servlet containers can be used - there's nothing Tomcat-specific in the codebase, but alternatives haven't been tested.

1. Estimated time to build : 2 minutes.
2. Desired structure : Similar (eventually) to the following, based on the "Advanced Configuration - Multiple Tomcat Instances" section of [this document](#).

```

<app-base>
|
+----/tomcat

```

(continues on next page)

(continued from previous page)

```

+---/CATALINA_HOME
|
|   +-----/app-manager -> ../vhosts/app-manager
|   +-----/app-manager.env
|   +-----/bin/catalina.sh
|   |       +-/startup.sh
|   +-----/client -> ../vhosts/client
|   +-----/client.env
|   +-----/client-direct -> ../vhosts/client-direct
|   +-----/client-direct.env
|   +-----/host.pkcs12
|   +-----/lib/catalina.jar
|   |       +-/servlet-api.jar
|   |       +-/tomcat7-websocket.jar
|   +-----/local_runner.sh
|   +-----/logs/
|   +-----/prepare.sh
|   +-----/procdir/
|   +-----/README.openssl
|   +-----/reset_all.sh
|   +-----/reset.sh
|   +-----/secure.env
|   +-----/start_all.sh
|   +-----/trustedkeystore.jks
|
+---/vhosts
|   +---/app-manager
|   |   +----/bin/tomcat-juli.jar
|   |   +----/conf
|   |   |       +-/Catalina/
|   |   |       +-/server.xml
|   |   +----/host.pkcs12 -> ../../CATALINA_HOME/host.pkcs12
|   |   +----/lib/
|   |   +----/trustedkeystore.jks -> ../../CATALINA_HOME/
->trustedkeystore.jks
|   |   +----/webapps/
|   |
|   +---/client
|   |   +---/bin/tomcat-juli.jar

```

1. `cd` <any empty temporary directory>
2. Download [Tomcat install scripts](#), retaining directory structure.
3. Run `linstall.sh` <tomcat source code> <tomcat destination> <tomcat version>
e.g. `linstall.sh /apps/src/apache-tomcat-7.0.56 /apps/tomcat 7.0.56`
4. `cd` <tomcat destination>/CATALINA_HOME and follow instructions in `README.openssl`

9.1.8 git

git (<https://git-scm.com/>) is a VCS (Version Control System).

Useful commands

Command	Operation
<code>git rev-parse --short HEAD</code>	Reveals the current commit hash
<code>git fetch</code>	Fetch (but don't merge) remote changes
<code>git diff --name-status HEAD...origin grep -P '(/sample\. \.gitignore \.xsd installation/components/.*?/index\.rst\$/appCtx\.config\. /pom\.xml \.sql)'</code>	Reveals remote-modified config files
<code>git difftool HEAD...origin -y -x "diff -sibv --suppress-common-lines -W 250" <file></code>	View remote-modified change
<code>git checkout FETCH_HEAD -- <file></code>	checkout the git fetched file
<code>git merge origin</code>	Merge git fetched changes
<code>git clean -dnx cut -c 14- grep -v '/\$'</code>	List of un-tracked files ¹

9.1.9 subversion

subversion (or svn) (<https://subversion.apache.org/>) is a VCS.

Useful commands

Command	Operation
<code>svn copy http://<svnserver>/svnrepo/client/trunk http://<svnserver>/svnrepo/client/tags/development/<yymmdd>_<git hash> -m "AP-Portal - client : Tag dev. update"</code>	Copies <i>trunk</i> to a <i>tag</i>
<code>svn diff --diff-cmd "/usr/bin/diff" --extensions "-yib --suppress-common-lines -W 250" client</code>	Show difference of local copy
<code>svn st -u</code>	What would be updated
<code>svn propget svn:ignore -R</code>	Ignored files/dirs

¹ Derived from stackoverflow.com

FAQ (Frequently Asked Question)

10.1 FAQs

10.1.1 How do I control user access to the portal?

For the `client` and `client-direct` components access is by default restricted to most areas, only information such as contact details and “about” information are unrestricted.

Both components share the ability to use a pre-populated user database (by means of an SQL script, e.g. `sample.users.sql` as a source of authentication (“*prepopulated user authentication*”), however if you wish to use a *site-specific* authentication mechanism (“*bespoke user authentication*”) then a corresponding *bespoke client* extension needs to be developed to provide or interact with the desired authentication mechanisms.

For other components, e.g. `app-manager`, `dose-response-manager` and `site-business`, access control is determined by a number of factors, such as those listed in *communication security*.

10.1.2 How do I adjust the?

See also:

The various *Maintenance* sections for common requests for altering configurable portal aspects.

11.1 Troubleshooting

Please see the following for assorted troubleshooting sources of information.

11.1.1 ApPredict problems

ApPredict invocation by app-manager fails

There could be a number of reasons for this happening, but assuming that the prerequisites are in place, i.e.

1. Files `local_runner.sh`, `prepare.sh` and `ApPredict.sh` (see *app-manager tools*) are working. I.e. they are in the right location, executable, and perform the right actions!
2. The location specified in `app-manager's src/spring/properties` file, `base.dir` property is valid, i.e. can be written to by the user under which the `app-manager` application is running.

... then it should be working!

See also:

Simulation job failed to run.

If it's not then the `base.dir` location can sometimes contain files generated by `local_runner.sh` which are useful for debugging such as `VRE_INFO.<app manager id>.xml` and `VRE_OUTPUT.<app manager id>`. These files are usually transitory in nature, i.e. they exist only for the duration of the simulation, and their content is normally persisted on successful `ApPredict` completion.

Warning: If there are failures to run then it may be necessary to manually delete the content of failed runs from the `base.dir` location.

11.1.2 Display problems

client display failing

If the display is failing following an update to AP-Portal then it could be symptomatic of legacy files being found in the browser's cache, thus preventing the latest operations being executed properly. See the following section on *Clearing a browser cache*


Clearing a browser cache

If the portal has been updated but there appears to be no change to what the user is seeing in their browser then it's possible that they are seeing the content from the browser's cache. In this case, then below is some general advice for forcing content to be removed from the browser cache.

IE (Internet Explorer) (based on ver. 11)

Tools -> Internet Options -> Browsing History -> Delete... -> make sure *Temporary Internet files and website files* and *Cookies and website data* are checked -> Delete.

Chrome (based on ver. 56.0.2924.87)

Ctrl+Shift+Del (or click on the Chrome button which looks like  then More tools -> Clear browsing data...) -> make sure *Cached images and files* is checked -> Clear browsing data.

Firefox (based on ver. 38.0.5)

Edit -> Preferences -> Advanced -> For *Cached Web Content* click the Clear Now button.

11.1.3 Java problems

This is probably most relevant with regard to *Tomcat* configuration.

Java has a number of performance and configuration options available which can be determined using sources such as the following

- For command-line *Java 7* args.
- Determining the defaults: `java -XX:+PrintFlagsFinal -version`.
- For *Java 7* Garbage Collection.
- 3rd party *Heap Space* explanation.
- *Stack Overflow* application memory usage.

See also:

Java environment setting.

Java memory use / OutOfMemoryError

If this error appears in the process of *log file analysis* then it may be necessary to increase the amount of memory available to Java, perhaps using the `JAVA_OPTS` environment var (or `CATALINA_OPTS` if specifically for *Tomcat*), e.g. `CATALINA_OPTS="-Xms1g -Xmx1g -Xss1g -XX:PermSize=256m -XX:MaxPermSize=256m"`. Having said that, it has been noted by some commentators that increasing the available memory may simply obscure temporarily a deeper problem such as memory leaking.

Java memory use / PermGen

It looks like *Maven* site building may fail on Java 7 due to PermGen space issues in some circumstances. Java 8 runs fine.

11.1.4 Log file analysis problems

These are a great source of information if the log level allows plenty of debugging information to appear. Generally the log level and location of the portal components is determined by the properties files (e.g. `src/properties/env.properties`, `src/properties/filter.properties`) when building components, or in the `<component>/webapps/<component_name>/WEB-INF/classes/log4j.xml` files when starting/running the component.

Apart from the component logging there is also the servlet container logging, e.g. *Tomcat*'s `catalina.out`, which may assist in problem solving.

NullPointerExceptions in site-business logs

If `AppManagerServiceImpl.java`'s `persistResults()` is generating `NullPointerExceptions`, this can happen because of [issue #23](#).

11.1.5 Server memory problems

Low free RAM

If you have a device which is low on RAM or doesn't manage buffering/caching well you may periodically need to flush it, e.g. `sync; echo 1 > /proc/sys/vm/drop_caches` (although please RTFM (Read The F!!!ing Manual) as to do so may cause issues in certain circumstances).

12.1 Glossary

assay group Refers to the grouping of different assays (of differing *assay levels*) which demonstrate similar characteristics (historically based on their accuracy) into a hierarchically organised representative group¹.

assay level Refers to the hierarchical organisation of individual assays (historically based on their accuracy) and the assignment of a corresponding assay level value¹.

assay value inheritance Refers to AP-Portal's mechanism for enabling IC50 (50% Inhibitory Concentration) values recorded by one assay to be inherited by another assay (based on their hierarchical ordering) as a substitute for missing measurements. For example, an IonWorks Barracuda® IC50 value for NaV1.5 being substituted into PatchXpress® assay data if no PatchXpress® NaV1.5 measurement existed¹.

back-end components Refers generally to the non- public-interfacing components (for example business-manager, app-manager, dose-response-manager, etc).

change checker Device which detects whether an existing simulation for a compound needs to be re-run for whatever reason, for example, if it detects different input IC50 values between the previous run request and the current run request (due to new assay data availability)¹.

dose-response data A generic term indicating the dose-response data (e.g. the individual dose-response points) associated with a *individual data* record.

emulator See *variability lookup tables*.

experimental data Refers to data derived from animal studies such as ventricular wedge assays.

front-end components Refers generally to the public-interfacing components, i.e. `client` and `client-direct`.

generic portal components Refers to the open source, publicly available portal components, e.g. `client`, `business-manager-api`, etc, which have generic (i.e. non- *site-specific*) processing.

individual data A generic term indicating the result of an assay experiment. There may be a number of *individual data* records for a compound, and for some systems these *individual data* records are aggregated into a *summary*

¹ `client` system only feature.

data record. Alternative names which may be encountered are “full curve” or “percent inhibition” data. An *individual data* record may also link to the detailed *dose-response data* from which it is derived.

input data gathering Refers to the process of directly calling `site-business` via a WS invocation in order to determine the pIC50 input data for a compound, i.e. it won’t run a simulation¹.

internationalised AP-Portal’s I18N capabilities.

local Very generally refers to objects on the portal’s host filesystem, the portal itself, or perhaps the wider *on-site* area.

pIC50 evaluation pIC50 evaluation is the technique by which AP-Portal determines the pIC50 value (or values) to use (**or reject, perhaps due to quality control**) in a simulation for each available ion channel and assay combination. There will be a number of pIC50 evaluators available, each representing a method of obtaining a pIC50 from the available data (e.g. perhaps both IC50 and IC20 (20% Inhibitory Concentration) values are available), organised hierarchically, with the expectation that the pIC50 value to use (or reject) will be derived from the first evaluator to provide (or reject) a pIC50.¹.

portal repository Refers to the portal’s generic, version-controlled source code repository at https://bitbucket.org/gef_work/ap_predict_online.

screening data Refers to data derived from ion channel screening device assays such as HTS (High Throughput Screening) machines, e.g. PatchXpress®.

site-specific Refers to bespoke configurations, settings or source code which is specific to the site which is hosting the portal, e.g. a site’s ion channel data processing code, a site’s user authentication code, or a site’s ion channels and assay configurations.

summary data A generic term indicating that *individual data* has been aggregated into a summary record somehow.

variability “*variability*” may be used interchangeably with “*spread*” and “*uncertainty*”! This term refers to ion channel variability, as explained in this [research article](#).

variability lookup tables In order to present *variability* results ApPredict requires the availability of large *lookup table* files. Since early 2018 these tables have been also been referred to as “Emulators”.

A

assay group, [97](#)
assay level, [97](#)
assay value inheritance, [97](#)

B

back-end components, [97](#)

C

change checker, [97](#)

D

dose-response data, [97](#)

E

emulator, [97](#)
experimental data, [97](#)

F

front-end components, [97](#)

G

generic portal components, [97](#)

I

individual data, [97](#)
input data gathering, [98](#)
internationalised, [98](#)

L

local, [98](#)

P

pIC50 evaluation, [98](#)
portal repository, [98](#)

S

screening data, [98](#)
site-specific, [98](#)

summary data, [98](#)

V

variability, [98](#)
variability lookup tables, [98](#)