URDMS Integration

The URDMS Integration project is a collection of NServiceBus (<http://nservicebus.com/>) services for coordinating site-provisioning, data collection approvals, document creation (for snapshots of the DMP) and publishing data to VIVO integrating with the open source Java Application (<http://sourceforge.net/projects/vivo/>). The two main workflows considered in the application are detailed below.

# Provisioning

## Provisioning services

The provisioning workflow is concerned with creating a SharePoint site collection for a particular project, adding users with the correct permissions, adding document artefacts and also notifying relevant parties along the way. It involves interactions between several services as detailed below.

### ProvisioningService

The provisioning service handles messages to create a sharepoint site which are sent by the web application when a user completes a data management plan or data deposit form. The initial message contains a project name, description and id as well as all the users that need to be given access to a SharePoint site collection and their respective roles. The provisioning service notifies the sharepoint (provisioning service) to create a sharepoint site. It also sends a message to the notification service which then emails the user that the request to provision a site has been received.

Once the sharepoint provisioning service has set up the sharepoint site, it informs the provisioning service which in turn messages the notification service to send another email as well as the view model updater service. The view model updater service is responsible for updating the state of the data management plan or data deposit to “provisioned”. This ends the provisioning workflow.

### ProvisioningService.SharePoint

As part of the provisioning workflow, the sharepoint provisioning service is responsible for communicating with the SharePoint installation via the Client Object Model to create a new site collection for a project, add users and allocate their access.

### NotificationService

The notification service is made up of several handlers for messages it is subscribed to. The service is responsible for sending out emails to notify users of the relevant steps within the provisioning workflow. The service uses ActionMailer.Net (standalone) to send emails while the templates are generated with the RazorEngine standalone (currently does not support layouts hence common heater/footer/css needs to be on each and every template).

The service will email all users listed in a project to notify them that a site is currently being provisioned for the submitted DMP/DD. Similarly, once the site is provisioned, all users within that message are sent an email to notify them that the site has been provisioned and they've been given access to it (the site URL is also sent).

### DocumentBuilderService

The document builder service is responsible for generating a snapshot of a DMP when a site is provisioned. Both PDF and XML versions are generated and then deposited in the newly provisioned SharePoint site using the client object model. All data is extracted directly from the web application database which means that any changes to the schema or content in the web application must be passed on to the repository method and final document outputs.

### ProvisioningService.ViewModelUpdater

The view model updater service for the provisioning workflow is fairly simple, it is only subscribed to one message. When that event is raised the view model updater will directly update the status of a DMP or DD in the web application database to DmpStatus.Provisioned for the respective project.

# Approvals

## Data collection approvals services

The data collection approval workflow is in charge of carrying a data collection through all approval steps until finally publishing the data collection by inserting a record into a VIVO database table (the VIVO application is then responsible for pushing the record to ANDS).

### Approvals.ApprovalService

The approval service is the main service in charge of the long running approval workflow. It contains a saga that handles all messages sent from the web application. The initial message when a researcher completes a data collection and confirms it is ready to be published commences the saga. The service then publishes a message for the view model updater to update the data collection status directly in the web application database.

The handlers follow this basic pattern for all of the following messages within the workflow until the final message is received to publish the collection, at which point the VIVO publisher service is sent a message. Once the response to that message is returned, that view model updater is alerted to make the final status change and the workflow is complete.

### Approvals.ViewModelUpdater

This service is very straight forward, it is only subscribed to an ApprovalStateChanged message which is published by the approval service. The message contains the new approval state of the data collection as well as the datetime of when it was changed and by whom. The service updates the web application database with this information.

### Approvals.VivoPublisher

The VIVO publisher service handles only one message, the ExportToVivo message published by the approval service. Upon receiving this message, the publisher service retrieves the data collection that has been approved from the web application database and then saves it into a VIVO schema appropriately. The service is NOT responsible for the final publishing of the data to ANDS, the VIVO application handles this portion.

# Installation/Set Up

* Create “Urdms.Approvals”, “Urdms.Approvals.Subscriptions”, “Urdms.Provisioning”, “Urdms.Provisioning.Subscriptions”, MS SQL databases.
* Set up Message Queuing
  + Go to “Control Panel” -> “Programs and Features”
  + Select “Turn Windows Features on or off”
  + Under “MicroSoft Message Queue (MSMQ) Server” select “MicroSoft Message Queue (MSMQ) Server Core”.
  + Restart your computer.
* The following MSMQ private queues are needed “urdms.approvals.approvalservice”, “urdms.approvals.approvalservice.timeouts”, “urdms.approvals.viewmodelupdater”, “urdms.documentbuilderservice”, “urdms.approvals.vivopublisher”, “urdms.notificationservice”, “urdms.provisioningservice”, “urdms.provisioningservice.timeouts”, “urdms.provisioningservice.sharepoint”, “urdms.provisioningservice.viewmodelupdater” and “urdms.error”. The application should create these for you when it is first run but if for some reason this did not occur, see the next item for how to add them manually.
* To add queues
  + Right click on “Computer” from the start menu or desktop and select “Manage”.
  + Open “Services and Applications” menu item
  + Open “Message Queuing” menu item
  + Right click on the “Private Queues” menu item and select “New” -> “Private Queue”.
  + Enter in the queue name and click OK.

# Implementation

* To run and debug the solution you will need to set some start up projects so all required services are up and running. Set the following start up projects; “Approvals.ApprovalService”, “Approvals.ViewModelUpdater”, “Approvals.VivoPublisher”, “DocumentBuilderService”, “NotificationService”, “ProvisioningService”, “ProvisioningService.SharePoint” and “ProvisioningService.ViewModelUpdater”. Now when you debug your solution you will see console windows for each service and output will inform you of the state of the service, queues, database, etc.
* Take note that the NServiceBus host is limited to 1 worker thread without a valid license. This may be enough to suit your needs but if it isn’t then you’ll need to acquire the appropriate licence.
* The document builder service was built using the ABC PDF library to generate PDFs (although it also generates XML documents from the data). The code to do so has been left in place however you’ll need to acquire a license (if you intend to keep using this service), include the library and uncomment the relevant areas of code. You may also need an xcopy command to copy the library to the bin/Debug folder, post-build. E.g.
  + xcopy "$(ProjectDir)..\lib\abcpdf\ABCpdf8-64.dll" "$(TargetDir)" /Y /C /D
* The final step in creating a Data Managerment Plan is the provisioning of a sharepoint site. Once again the code has been commented and left in place in case you also choose a sharepoint solution, you can choose to reuse this code or implement another solution (or do nothing) at this stage.

# Deployment

To Deploy the NServiceBus services to test or production:

1. Build the solution in Visual Studio in Release mode.
2. Run the PackageForDeployment.bat file in the root of the solution folder.
3. Zip the folder created with today's date/time in the \_Deploy folder and send it to whoever can deploy to the required environment.
4. The person with permissions to deploy should then unzip the file on the target server and running the Deploy.bat file providing the username and password of the service account used to run the services   
   i.e., Deploy.bat service\_account service\_password
5. This will install the services and start them automatically.

# App.config settings

* All NServiceBus endpoints have already been configured appropriately although some extra details will need to be entered for some of the services.
* All App.config files have transforms for the CI, Test and Prod servers.
* You’ll need to update database connections, email addresses, URLs and mail servers to appropriate values.