NetQuarry

The Enterprise Application Software Development Platform for Microsoft.NET

IssueTrak Tutorial – 1 – The Platform

Table of Contents

Purpose of this document

This document is a tutorial that explains how to build an application using the NetQuarry Enterprise Application Platform. The document is split into three parts.

Part 1 – Explains how the NetQuarry Platform works and how the platform should be installed and configured.

Part 2 – Takes you through a set of steps to create a functional bug tracking application. The steps will be deal entirely with manipulating metadata to create the IssueTrak application.

Part 3 – Enhances the basic functionality of the IssueTrak application by adding complex business rules with C# extensions, creating scheduled tasks and debugging.

NetQuarry Overview

NetQuarry is an enterprise application software platform for Microsoft.NET. It is comprised of pre-built software, metadata and tools designed to help professional software teams develop significantly better enterprise and hosted business applications. Developers using NetQuarry will deliver everything faster (less time and manpower equals less cost):

- Rapid Prototypes (in as little as one day)
- Initial functional applications that perform like highly mature ISV solutions
- Subsequent versions and improvements without major rewrites.
- Agile response to the reality of changing requirements

Key Benefits

- Rapid results/development cycle
- High product functionality with low project execution risk
- Low ownership cost
- Low maintenance cost
- Low hardware cost
- Rich functionality for customization

NetQuarry provides a foundation of functionality common to virtually all enterprise class applications The common functionality includes data binding, database virtualization, OS-virtualization, searching, filtering, user interface (UI), security / permissioning, and numerous other commonly required capabilities. In each case the features are designed with great attention to the enterprise-class foundation, and with the knowledge of what custom applications, the developers *and* users of those applications really need.

Agility

Today’s dynamic business environments demand a great deal from developers and their managers. Agile, iterative development allows developers to incorporate frequent feedback from end users, realizing that successful applications are constantly refined, changing rapidly to meet the needs of the enterprise.

Making agile development successful requires advanced software infrastructure and disciplined methodologies. Infrastructure should be designed to facilitate iterative development, producing quality applications.

NetQuarry and Agile Principles

The following basic principles (loosely adapted from the agile development manifesto)

- The highest priority is to satisfy the customer through early and continuous delivery of valuable software. Building infrastructure is a requirement to basic delivery of software, however, customers cannot "see" infrastructure. NetQuarry removes this requirement completely.
- An agile development platform and approach allows developers to welcome changing requirements, even late in development. NetQuarry allows developers to harness change for the customer's competitive advantage.
- Working software is the primary measure of progress. NetQuarry allows developers to deliver *working* software frequently, from a couple of days to a couple of months.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to *technical excellence* and good design enhances agility. NetQuarry is the result of proven good design and quality.
- Simplicity—the art of maximizing the amount of work not done—is essential.

Extensibility

A key concept of the NetQuarry product is its extensibility. Although from past experience we understand most of the commonly required features; we also know that we cannot anticipate or accommodate all custom application needs. For this reason NetQuarry provides a comprehensive extension model. This model allows extension to the basic functionality through creation of .NET-compatible components that are configured into the system to respond to system events. This model makes NetQuarry an excellent platform not just for new application development, but also for **Enterprise Application Integration (EAI)**. Using NetQuarry, multiple external applications can be integrated into a single front-end. In addition, because of the meta-data used to specify Business Objects in NetQuarry, **NetQuarry can provide SOAP and Service Oriented Architecture (SOA) definitions.**

Key Features

- A rich object-data source-mapping layer that is declared by the developer using tools provided by NetQuarry. Object definitions are stored in the EAP metadata.
- A detailed event model and event framework that allows the developer to write domain specific business rules in fully independent and contained

classes. The developer does not need to inject utility code (like code-generators) into their classes. Events can be handled by a chain of classes and can be reused across multiple Objects. The existence of the event handlers (called "extensions") is stored in the EAP repository.

- A feature rich user interface layer that includes integrated binding to the underlying business objects and total customization. The EAP stores the UI definitions in the repository and dynamically renders the user interface at runtime. Most changes can be applied by configuration (declarative programming) while the application is running. Like all of the other meta-data objects, they are persisted into the repository along developer specified boundaries ("modules").
- Enterprise class services including:
- Role-base Security
- Authentication
- Searching/Filtering
- Internationalization
- "First time achievable scalability"
- Integrated task scheduling

The EAP is a declarative framework that stores relevant application information in a configurable, deployable meta-data repository and allows domain specific extensions to the application to be developed in any .NET language. You reuse and extend the EAP in the same way that the EAP extends the .NET framework.

Architecture

The NetQuarry Enterprise Application Platform (EAP) allows you to build and deploy n-tiered, enterprise quality applications. The EAP sits on the Microsoft .NET common language runtime and provides rich functionality at all layers of an application.

The NetQuarry EAP does not generate code, rather, it uses the NetQuarry Core Runtime, custom configuration, metadata, and developer coded extensions to construct world class, quality applications.

The application is divided into four logical layers: Presentation, Application, Data Access, and Services. Each layer supports customization and configuration at every level.

Presentation Layer	Application Layer	Database Layer	Services Layer
Datasheets	Role-based Security	Database independence	Custom WebService provider
Web Forms	Robust, simple extension model	Object Relational Mapping	Inbound/Outbound Email handling
Rich Filtering, Sorting	Rich Validation	Enterprise Scalability	Alert management

Client and Server-side field validation	Internationalization Support	Multi-Data Source support	Task scheduling
Custom Layouts and Template support	Scheduled Task Support	Non-relational data support	Data import/export management

All of the NetQuarry runtime functionality is fully managed code and 100% .NET.

The presentation layer is a set of reusable ASP.NET based user interface elements. There is full support for custom ASP.NET pages, either bound to the underlying NetQuarry Data Mapping objects or completely standalone. The presentation layer is a first class consumer of the application layer interface.

The application layer provides a fully componentized, loosely coupled, and reusable extension model that allows the application developer to add their own domain-specific business logic without wading through *any* generated or platform code. Additionally, the application layer provides a single, consistent interface for accessing business logic and data access. This same interface is used by the platform to expose your custom application-layer objects as WebServices, allowing you the same interface via SOAP that the presentation layer uses to provide an ASP.NET end user experience.

The platform also supports a type-safe extension model using platform generated .Net generic (template) classes.

The data access layer uses ADO.NET to access data sources across your enterprise, regardless of the underlying data store; relational, WebService-based, or procedural. The Database layer provides true data source independence and rich Object Relational Mapping, all with enterprise scalability.

Finally, the services layer provides rich runtime services, such as email processing, alert management, and task-scheduling. In addition, applications created using NetQuarry have their business objects automatically exposed as fully typed WebServices to the external world.

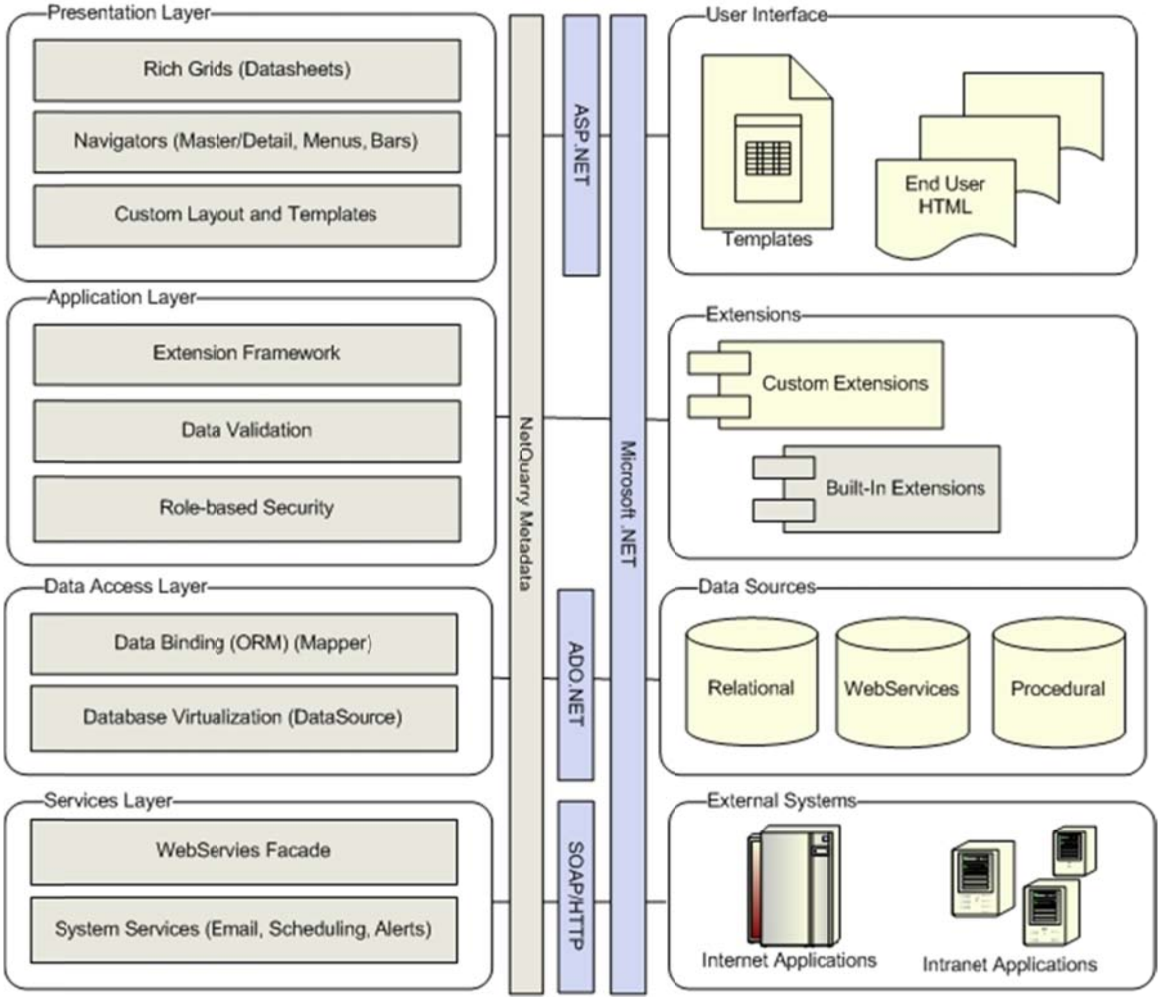
How it works

When trying to understand NetQuarry, it is helpful to understand several aspects of the product. First, what is the physical structure and content of the software? Second, what sorts of users interact with it and what does that look like? Third, what are the most important components and how do they interact?

The physical structure

NetQuarry is a complete set of Microsoft .NET libraries, executable applications, web pages, and meta-data. The entire platform has been written in 100% managed .NET code. The core platform libraries expose a set of managed classes that drive the application using meta-data as a primary means of configuration.

How the pieces fit together



Users

The platform has two basic faces for each of the two main user groups. Application developers interact with the NetQuarry Studio, a Windows .NET application that configures the runtime application's meta-data. End-Users interact with an ASP.NET web application (running in the context of IIS) that uses the configured meta-data to build the user interface, manipulate and format data from any number of data sources (both relational and not), control authentication and permissions, and manage custom business logic.

Application Developers

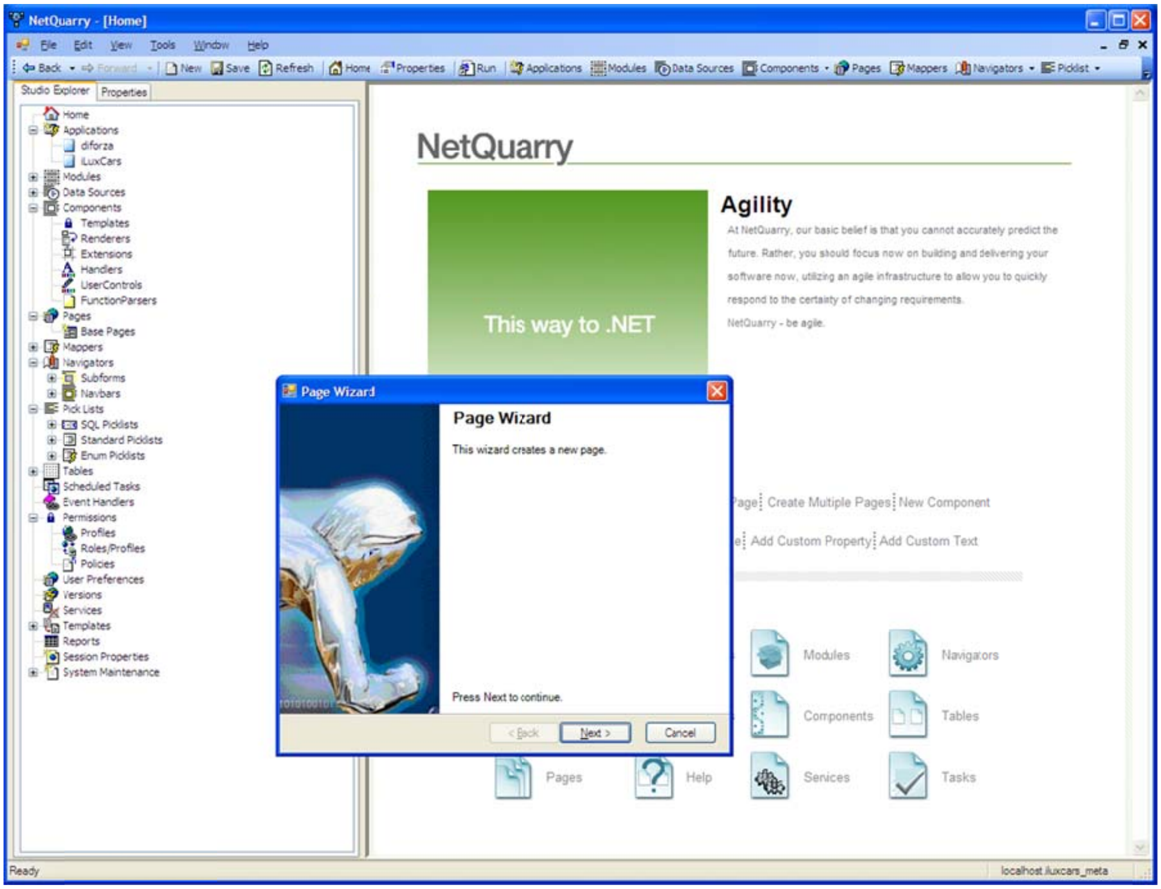
Application Developers build applications by doing one of three primary tasks. First, a developer develops or analyzes the data that is to be used in the application. Second, the developer uses NetQuarry Studio to manipulate meta-data that describes to the NetQuarry runtime how the end application will look and manage the application's data. Third, any custom business logic is written using any .NET language and IDE (e.g. Visual Studio .NET) to produce focused application extensions.

Data Sources

Business applications depend on data. Increasingly, this data comes from multiple data "sources" across the enterprise. These sources of data may exist in relational databases such as Oracle, Microsoft SQL Server, or IBM DB2. Some of the sources of data, however, exist as Web Services, XML documents, or remote procedure calls. NetQuarry makes aggregating disparate data into a single, functional user interface simple and fast.

NetQuarry Studio

To describe an application to the NetQuarry runtime the developer interacts with the NetQuarry Studio. The Studio is a Windows application that manipulates the meta-data objects and guides the developer through the creation and management of these objects.



Agility

At NetQuarry, our basic belief is that you cannot accurately predict the future. Rather, you should focus now on building and delivering your software now, utilizing an agile infrastructure to allow you to quickly respond to the certainty of changing requirements.

NetQuarry - be agile.

Page Wizard

This wizard creates a new page.

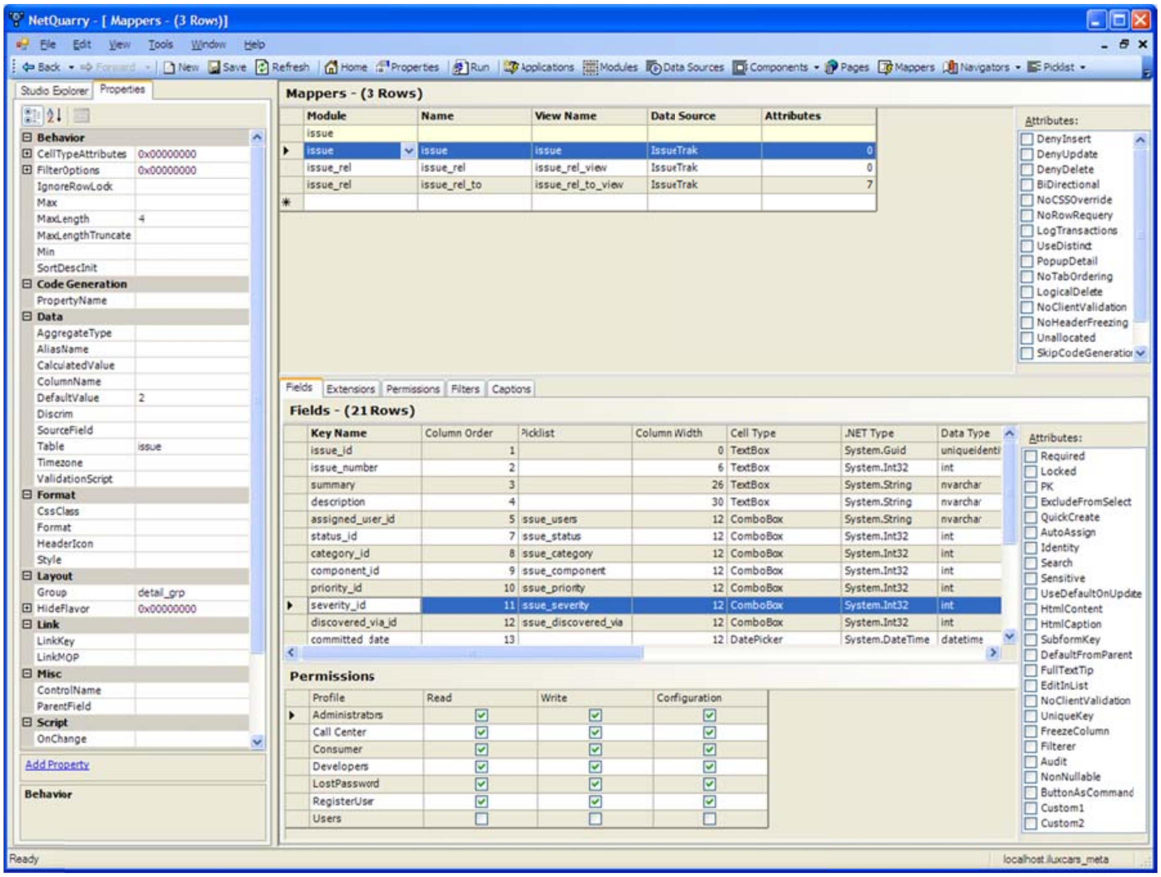
Press Next to continue.

NetQuarry Studio running Page Wizard

Task Wizards

Most of the major tasks related to metadata configuration have associated task wizards to guide the application developer through the process of creating objects.

The rest of the studio works in a familiar way to Visual Studio users, all objects support a set of properties specific to that object and the properties are manipulated via a familiar property sheet interface.



⊟ **Behavior**	
⊞ CellTypeAttributes	0x00000000
⊞ FilterOptions	0x00000000
IgnoreRowLock	
Max	
MaxLength	4
MaxLengthTruncate	
Min	
SortDescInit	
⊟ **Code Generation**	
PropertyName	
⊟ **Data**	
AggregateType	
AliasName	
CalculatedValue	
ColumnName	
DefaultValue	2
Discrim	
SourceField	
Table	issue
Timezone	
ValidationScript	
⊟ **Format**	
CssClass	
Format	
HeaderIcon	
Style	
⊟ **Layout**	
Group	detail_grp
⊞ HideFlavor	0x00000000
⊟ **Link**	
LinkKey	
LinkMOP	
⊟ **Misc**	
ControlName	
ParentField	
⊟ **Script**	
OnChange	

Behavior

Mappers - (3 Rows)

Module	Name	View Name	Data Source	Attributes
issue				
issue	issue	issue	IssueTrak	0
issue_rel	issue_rel	issue_rel_view	IssueTrak	0
issue_rel	issue_rel_to	issue_rel_to_view	IssueTrak	7

Attributes:
☐ DenyInsert
☐ DenyUpdate
☐ DenyDelete
☐ BiDirectional
☐ NoCSSOverride
☐ NoRowRequery
☐ LogTransactions
☐ UseDistinct
☐ PopupDetail
☐ NoTabOrdering
☐ LogicalDelete
☐ NoClientValidation
☐ NoHeaderFreezing
☐ Unallocated
☐ SkipCodeGeneratio

Fields - (21 Rows)

Key Name	Column Order	Picklist	Column Width	Cell Type	.NET Type	Data Type
issue_id	1		0	TextBox	System.Guid	uniqueidenti
issue_number	2		6	TextBox	System.Int32	int
summary	3		26	TextBox	System.String	nvarchar
description	4		30	TextBox	System.String	nvarchar
assigned_user_id	5	ssue_users	12	ComboBox	System.String	nvarchar
status_id	7	ssue_status	12	ComboBox	System.Int32	int
category_id	8	ssue_category	12	ComboBox	System.Int32	int
component_id	9	ssue_component	12	ComboBox	System.Int32	int
priority_id	10	ssue_priority	12	ComboBox	System.Int32	int
severity_id	11	ssue_severity	12	ComboBox	System.Int32	int
discovered_via_id	12	ssue_discovered_via	12	ComboBox	System.Int32	int
committed_date	13		12	DatePicker	System.DateTime	datetime

Attributes:
☐ Required
☐ Locked
☐ PK
☐ ExcludeFromSelect
☐ QuickCreate
☐ AutoAssign
☐ Identity
☐ Search
☐ Sensitive
☐ UseDefaultOnUpdate
☐ HtmlContent
☐ HtmlCaption
☐ SubformKey
☐ DefaultFromParent
☐ FullTextTip
☐ EditInList
☐ NoClientValidation
☐ UniqueKey
☐ FreezeColumn
☐ Filterer
☐ Audit
☐ NonNullable
☐ ButtonAsCommand
☐ Custom1
☐ Custom2

Permissions

Profile	Read	Write	Configuration
Administrators	☑	☑	☑
Call Center	☑	☑	☑
Consumer	☑	☑	☑
Developers	☑	☑	☑
LostPassword	☑	☑	☑
RegisterUser	☑	☑	☑
Users	☐	☐	☐

NetQuarry Studio showing Mapper View

Extensions

Extensions are a key concept of the NetQuarry platform. Extensions are libraries that receive events from the running application objects. For example, when a record in a page is changed, the Mapper representing that data fires update events (notifications) through all of the extensions configured (via metadata) on that object.

This is a powerful programming pattern that is extremely simple for application developers, even those unfamiliar with Microsoft .NET to quickly grasp and master.

To create an extension, the developer runs a wizard from the Studio that allows them to specify the name, location, language (C#, VB.NET), and which objects they want to attach the extension.

Some objects support the idea of "global" extensions (e.g. Mappers). A global extension is automatically attached to all objects of a particular class. This is particularly useful for functionality such as auditing and summary information handling.

Extensions must implement a known interface (NetQuarry.IExtension) or derive from a provided base class. For example, to build an extension that handles events on a Mapper (e.g. Update, Insert, Delete), the following code (generated via the extension wizard) provides the template.

To make extension implementation easier, NetQuarry provides TypedMappers and TypedExtensions based on TypedMappers. TypedMappers derive from mapper-specific base class generics (templates) generated by the platform. TypedExtensions use TypedMappers. TypedExtensions, using TypedMappers provide a type-safe way to manipulate mapper objects.

C#

```
using System;
using NetQuarry;        //--- include the NetQuarry objects
using NetQuarry.Data;   //--- include the NetQuarry data objects

namespace IssueTrak.Extensions
{
   /// <summary>
   ///
   /// </summary>
   public class XExt : MapperExtensionKernel
   {
      public XExt() {}

      public override void RowBeforeUpdate(IMapper sender,
                                               NQEventArgs e)
      {
         IField    fld = sender.Fields["forecast_discount_amount"];

         if (System.Convert.ToInt32(fld.Value) > 10)
            fld.Value = 10;
      }
   }
}
```

Handling business logic using this methodology is both simple and robust. Extensions can be reused across multiple objects, they can be debugged interactively using Visual Studio, and they hide the complexity of the infrastructure from the application developer allowing them to concentrate on the task of building business applications.

Platform Components

The common functionality includes data binding, database virtualization, OS-virtualization, searching, filtering, user interface (UI), security / permissioning, and numerous other capabilities. The platform is comprehensive and extendable. Most of the objects have both the capability to fire events through extensions and be configured via metadata.

DataSources (IDatabase)

As mentioned above, enterprise data may come from multiple data “sources” across the enterprise. These sources of data may exist in relational databases such as Oracle, Microsoft SQL Server, or IBM DB2. Some of the sources of data, however, exist as Web Services. A DataSource is an important, first class object in the NetQuarry platform. A DataSource is specified by name (and properties) in metadata, and the physical connection to it is specified in the web.config file.

NetQuarry uses a DatabaseProviderFactory pattern to create the correct connection type for each database. The provider pattern allows the data access code to treat different source types the same because they are all subclasses of the same base class. As far as the developer is concerned, the physical source of data is mostly immaterial.

Mappers

A Mapper is the primary means of mapping object-oriented programming objects to data sources managed by relational databases or other sources of data. A Mapper provides 2-way (Read and Write) access to the underlying Data Source.

There are several subclasses of the Mapper object. The most important are the MapperDatasheet and MapperDetail. Each provides an implementation that uses the data in the Mapper to render a view of the data using HTML.

Application (IAppContext)

The Application object holds information that is used throughout each page (or server) request. The Application is available to all objects, most importantly the extensions. The Application object is created as the first object in any request, and when possible it is cached for performance purposes.

The Application object handles connecting to the authentication provider, authenticating the user, checking object permissions for that user, storing user (UserContext) information, holding the list of available NetQuarry.PageInfo objects, and connecting to the metadata (Repository) data source. It also has a collection of DataSources specified in metadata.

More than one Application object may be specified in metadata, and connecting to an application allows the user to see more than one “view” of an implemented NetQuarry application (more on this in the security sections).

Pages and Templates

A Page is the smallest object that renders user interface. Some pages display data from a Mapper, while others allow the user to perform a quick search, quick creation of an object, or perform some sort of navigation. A Page can inherit from a Base Page.

A Template is a code-based layout with one or more regions that can host other UI elements. The regions are called Slots and the elements they host are called Page Elements. Page Elements are specified in the Page Element subform in the studio and each is assigned to a Slot. Different templates provide Slots.

Properties and TextItems

Many NetQuarry platform objects support a collection of custom Properties. Properties are fields of information that varies between objects and object instances. Most properties are loaded from the metadata when the object is created. A property that is loaded from metadata has an associated parameter object (a metadata only object) defined in the metadata. Properties can be added at runtime or design time. At design time, parameters form the definition for the NetQuarry studio's property sheet.

TextItems are similar to properties but hold only strings. Additionally, each TextItem object is specific to a locale (or culture). At runtime, Textitems are loaded for the current user's culture. If there is no string for the type of text for the current culture, the default culture is used. All strings used in the platform are represented as TextItems. This provides a simple way to support multiple user languages (cultures) on a single site, with a single codebase.

Logging

The platform is proactively logging a rich set of data to help the developer diagnose problems during development and runtime. Additionally, the developer can use the built-in object called DevLog to add messages to the log.

Security and Authentication

NetQuarry supports application security using a powerful, replaceable set of authentication and permission patterns. Depending on the needs of your application and the capabilities of your enterprise infrastructure, NetQuarry can support several different methods of user authentication.

Authentication Providers (IAuthenticationProvider)

An authentication provider implements the IAuthenticationProvider interface and handles the Authenticate method. NetQuarry ships with three providers, a Generic provider, a Windows based provider (for Active Directory or NTLM authentication), and a database provider.

Roles and Profiles

After authentication, the platform asks the authentication provider for a list of directory (ADS, NTLM, Database) roles for which the user is a member. These directory roles are mapped to runtime "profiles" which assign all runtime objects permissions. By default, an object is available to all users. The mapping of directory roles to profiles is done using the NetQuarry studio.

At runtime, the platform calls IAppContext.HasPermission to determine if an object is available to the current user.

Permissions

Most objects use the NetQuarry.ObjectPermissions enumeration to specify security for that object. The NetQuarry.ObjectPermissions enumeration is a superset of all possible system permissions (and is extensible). All attributes of the enumeration don't apply to every object. For example, NavigationTargets only support the "Read" permission bit.

Policies

For permissions that cannot be expressed as NetQuarry.ObjectPermissions bits, you can define a custom “policy.” A policy is simply a named permission that is granted to a set of profiles. This is a powerful and handy method for building an application with specific permissions for your users.

For example, one could define a policy that grants a set of profiles the right to mass email to the contents of a list of data. Then, before adding the custom MapperCommand that supported this functionality, the developer would ask the application if the current user had permission to the policy.

UserContext

After authentication, the UserContext object holds the current user identity and a collection of profiles. A UserContext object can re-create itself from a token securely. The IAppContext object holds the current UserContext.

Navigators

A navigator is an object that holds one or more “targets” (or links) that provide the end user with a way to navigate from one object to another. The most common types of navigators are subforms and navbars. Pages that have subforms have either a set of tabs or lists that represent related data. A Navbar works like a top-level menu and provides links to the top-level pages in the application.

PickLists

A PickList represents a domain specific list of values that can be used to translate the key of an item (typically a foreign key in a database table) to a description. All Mapper.Field objects have a PickList property.

PickLists are of one of three types:

- A “standard” PickList. This type of list is loaded from the metadata and supports the descriptive text in the current user’s culture (language).
- A “SQL” PickList. This type of list is loaded from one of the databases (can be the an operational database or the repository) and can contain 1 to 3 columns of data.
- An “Enumeration” PickList. This list is a 2 column PickList loaded from a string type name as its source.

PickLists have one or more of the following attributes:

Cache	This list should be cached.
HasDiscrim	This list uses a discriminator to break items into sets.
LimitToList	This list has at least 2 columns and the values must come from the list.
HideUnknownItems	The list should hide unknown items.
MarkUnknownItems	The list should mark unknown items.
StoreAltText	The list stores the item's Alternate Key Text column in the DB and

	uses it as it's ID.
StoreAltInt	The list stores the item's Alternate Key Int column in the DB and uses it as it's ID.
StoreItemName	The list stores the item's Name column in the DB and uses it as it's ID.

For the most part, PickLists tend to have the LimitToList attribute set. This attribute means that the list has a key (hidden) and text (displayed). For example, if you had a column in a database that held an integer ID for a customer status you might create a PickList like this:

1	Active
0	Inactive

This list could be of type “standard” and the text (Active, Inactive) would be localizable.

Note that PickLists are most often used with dropdown lists (comboboxes), lists, and radio buttons (controls offering the user a set of possible selections), but PickLists work with any cell type. For example, if a field uses a read-only TextBox, a PickList can still be used to resolve a stored value to its corresponding display text. The use of PickLists for this type of functionality is entirely reasonable providing that the number of items in the PickList is not excessive.

For performance and user-interface quality reasons, you should generally avoid having more than a few hundred items in a PickList.

For more information about the API objects see http://help.netquarry.com

NetQuarry Metadata

How the Metadata is stored

An application created with the NetQuarry platform uses metadata as the definition or description of the runtime application. At both design and run-time this metadata is stored in a relational database (e.g. SQL Server, Oracle, MySQL). The NetQuarry Studio reads and writes data to this operational store.

However, for several reasons, mainly the fact that a relational database is not a convenient format for source code control, the permanent format of the metadata is not a relational database. Instead, for configuration purposes, the "truth" of the metadata is stored in human readable, and easily diff-able, text format. This meta-data should be treated like code and stored in a version control system.

When editing the metadata, the tools allow the developer to segment sections of the metadata into groupings defined either by a "module" or an "application." The metadata can then be saved to the file system and checked into a configuration management system. Creating a current snapshot of the metadata is accomplished quickly and simply by loading the application module files into an empty metadata schema via a command line database initialization tool.

The segmentation of the metadata allows multiple developers to effectively work on a single application, ensuring that metadata changes can be handled by the configuration management system at the same level as source code.

While editing, the studio tracks changes to each module and allows the developer to export only those modules that have been changed.

Command Line Tools

For convenience and scripting purposes, it is generally recommended that you use the command line interface to save and load metadata. The tool (called MetaUtil) takes 4 parameters and will save an entire application's metadata to a single location:

Example:

```
call %NQBIN%"\EAP.Tools.MetaUtil.EXE" -save .
"Provider=SQLOLEDB;Data Source=.;Initial
Catalog=issues_meta;Integrated Security=SSPI" IssueTrak
```

Note that this assumes that you have defined an environment variable that points to the install/bin folder.

Prerequisites

Software required

NetQuarry requires the following:

- Microsoft Windows XP SP3 (recommended Windows 7 SP1 or later) or Microsoft Windows Server 2003 (Service Pack 1)
- Microsoft SQL Server 2000 (recommended 2008 or later)
- Microsoft .NET Framework Version 2.0 (recommended Version 3.5 or later)
- Visual Studio 2008.
- IIS Web Services. Exact version depends on installed operating system.
- Microsoft Report Viewer 2010
- NetQuarry Platform

Prerequisite guidelines

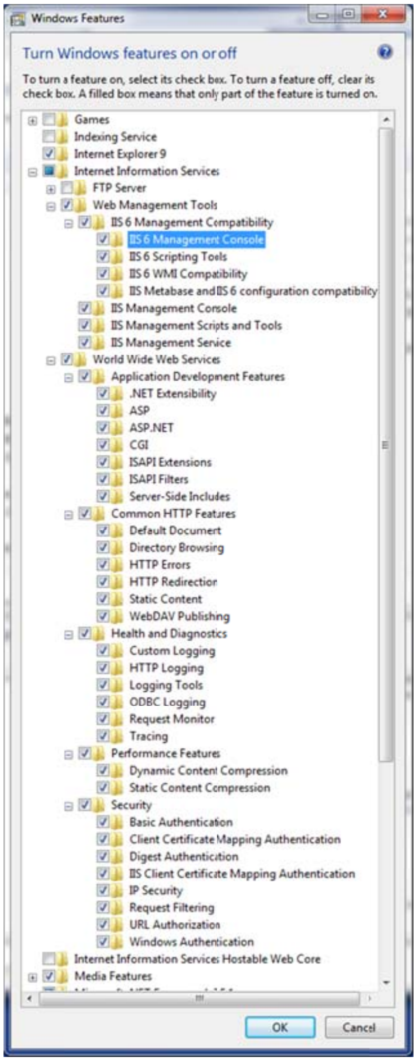
SQL Server Installation

When configuring your local installation of SQL Server, we recommend you assign the SQL Server service authentication to operate as the Network Service Account (NT AUTHORITY\NETWORK SERVICE).

The tutorial is written assuming that you have installed SQL Server in Mixed or Integrated authentication mode. All examples and scripts assume integrated security.

IIS Setup

When configuring IIS for versions of IIS 7 or greater, you also have to include in the setup the IIS 6 Management Compatibility Options



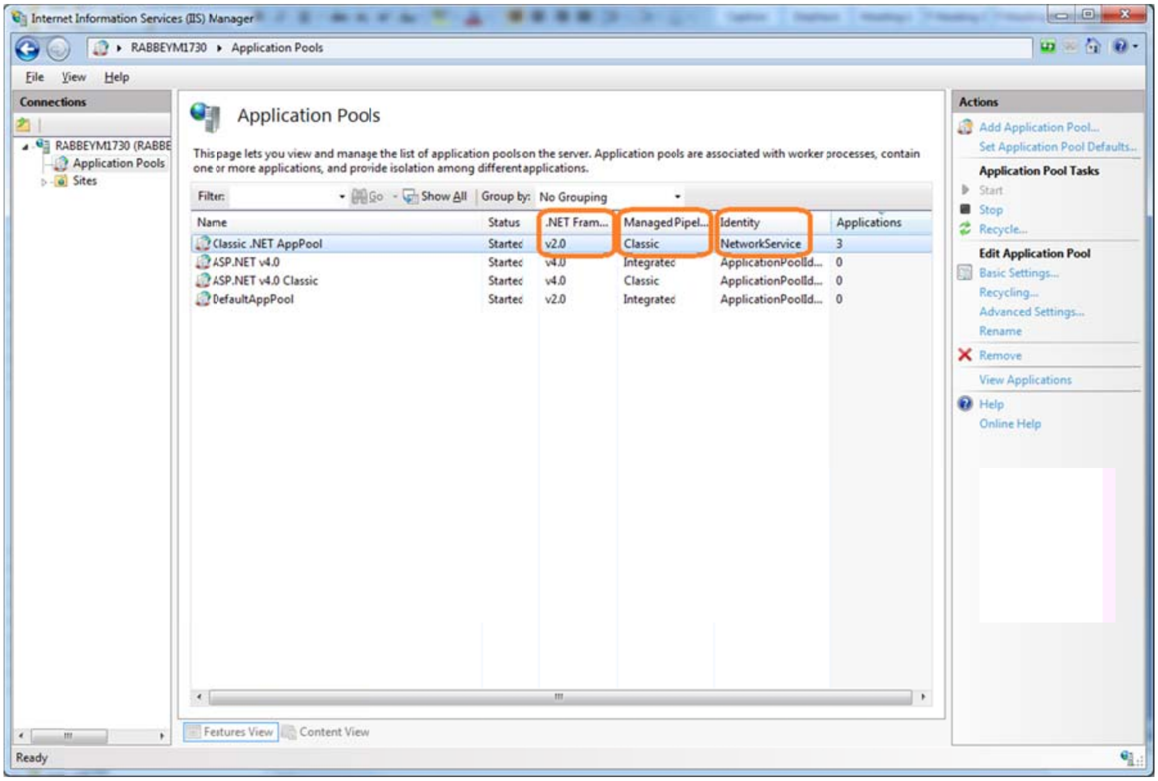
Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- □ Games
- □ Indexing Service
- ☑ Internet Explorer 9
- ■ Internet Information Services
  - □ FTP Server
  - ☑ Web Management Tools
    - ☑ IIS 6 Management Compatibility
      - ☑ IIS 6 Management Console
      - ☑ IIS 6 Scripting Tools
      - ☑ IIS 6 WMI Compatibility
      - ☑ IIS Metabase and IIS 6 configuration compatibility
    - ☑ IIS Management Console
    - ☑ IIS Management Scripts and Tools
    - ☑ IIS Management Service
  - ☑ World Wide Web Services
    - ☑ Application Development Features
      - ☑ .NET Extensibility
      - ☑ ASP
      - ☑ ASP.NET
      - ☑ CGI
      - ☑ ISAPI Extensions
      - ☑ ISAPI Filters
      - ☑ Server-Side Includes
    - ☑ Common HTTP Features
      - ☑ Default Document
      - ☑ Directory Browsing
      - ☑ HTTP Errors
      - ☑ HTTP Redirection
      - ☑ Static Content
      - ☑ WebDAV Publishing
    - ☑ Health and Diagnostics
      - ☑ Custom Logging
      - ☑ HTTP Logging
      - ☑ Logging Tools
      - ☑ ODBC Logging
      - ☑ Request Monitor
      - ☑ Tracing
    - ☑ Performance Features
      - ☑ Dynamic Content Compression
      - ☑ Static Content Compression
    - ☑ Security
      - ☑ Basic Authentication
      - ☑ Client Certificate Mapping Authentication
      - ☑ Digest Authentication
      - ☑ IIS Client Certificate Mapping Authentication
      - ☑ IP Security
      - ☑ Request Filtering
      - ☑ URL Authorization
      - ☑ Windows Authentication
- □ Internet Information Services Hostable Web Core
- ☑ Media Features

The NetQuarry platform is required to run under the "Classic" rather than "Integrated" pipeline mode. You may choose any application pool under which to run the NetQuarry platform as long as it is set to "Classic" managed pipeline mode. You may also create a new application pool for installing the NetQuarry platform, as long as the .Net version is set to 4.0 and the "Classic" managed pipeline mode.

For the application pool you have chosen to run the NetQuarry platform, also set the Identity to use the same Network Service account you configured for the SQL Server service logon.

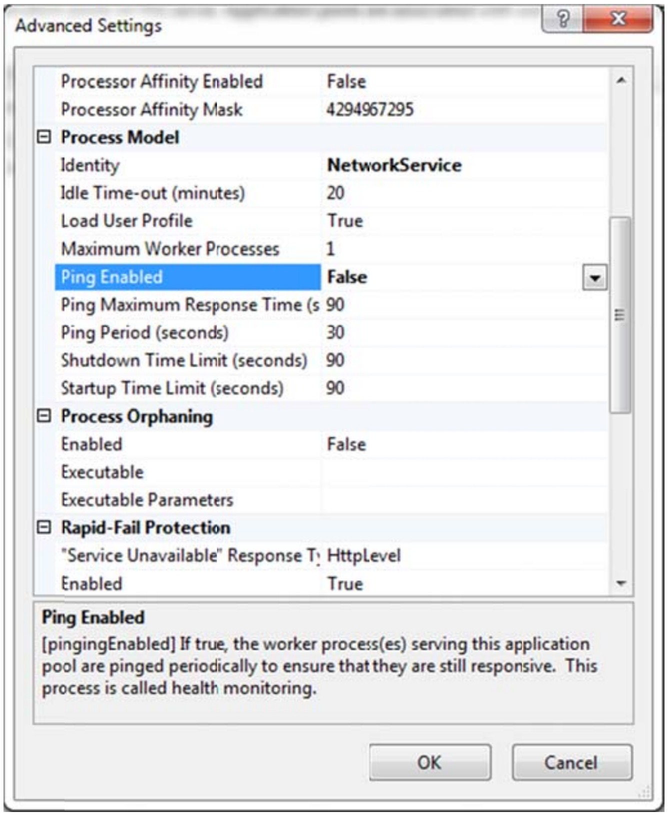


Application Pools

This page lets you view and manage the list of application poolson the server. Application pools are associated with worker processes, contain one or more applications, and provide isolation among differentapplications.

Name	Status	.NET Fram...	Managed Pipel...	Identity	Applications
Classic .NET AppPool	Startec	v2.0	Classic	NetworkService	3
ASP.NET v4.0	Startec	v4.0	Integratec	ApplicationPoolId...	0
ASP.NET v4.0 Classic	Startec	v4.0	Classic	ApplicationPoolId...	0
DefaultAppPool	Startec	v2.0	Integratec	ApplicationPoolId...	0

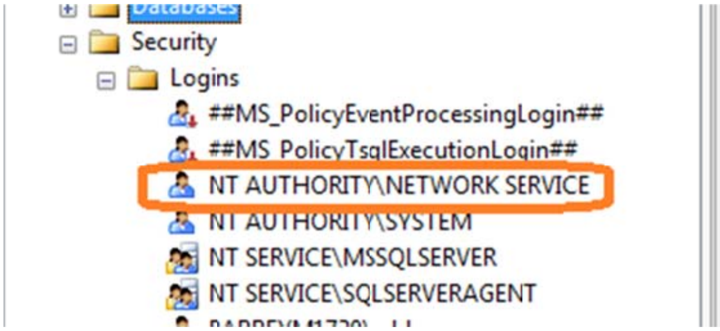
Set the identity of the application pool via the Advanced Settings of the application pool. Also, in the same property dialog. Under Process Model. Set Ping Enabled = false. This turns off IIS health checking monitor which allows you to debug your extension code without having the worker process resetting due to timeout.



Processor Affinity Enabled	False
Processor Affinity Mask	4294967295
⊟ **Process Model**	
Identity	**NetworkService**
Idle Time-out (minutes)	20
Load User Profile	True
Maximum Worker Processes	1
Ping Enabled	**False**
Ping Maximum Response Time (s	90
Ping Period (seconds)	30
Shutdown Time Limit (seconds)	90
Startup Time Limit (seconds)	90
⊟ **Process Orphaning**	
Enabled	False
Executable	
Executable Parameters	
⊟ **Rapid-Fail Protection**	
"Service Unavailable" Response T	HttpLevel
Enabled	True

Ping Enabled
[pingingEnabled] If true, the worker process(es) serving this application pool are pinged periodically to ensure that they are still responsive. This process is called health monitoring.

Using the Network Service users as the identity logon for the application pool is suggested, but you may use any identity for the application pool. However, if you do use a different identity, you should ensure that the same user/identity is added as a logon right to your SQL Server

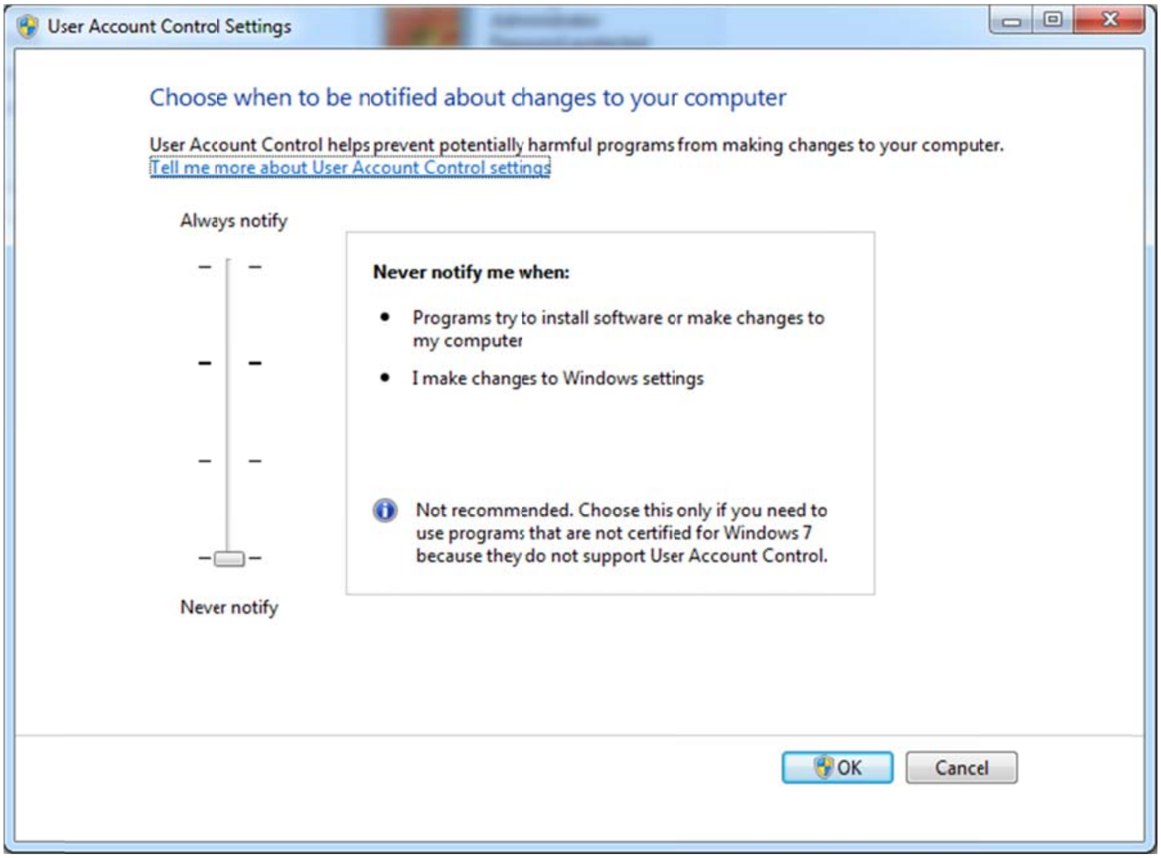


Databases
Security
Logins
##MS_PolicyEventProcessingLogin##
##MS_PolicyTsqlExecutionLogin##
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
NT SERVICE\MSSQLSERVER
NT SERVICE\SQLSERVERAGENT

And change the database scripts for database creation to add your own identity as a role member to the databases.

User Access Control (UAC)

We recommend that on Windows 7 platforms that you set the UAC notification level to "Never Notify"



User Account Control Settings

Choose when to be notified about changes to your computer

User Account Control helps prevent potentially harmful programs from making changes to your computer.
Tell me more about User Account Control settings

Never notify me when:

- Programs try to install software or make changes to my computer
- I make changes to Windows settings

Not recommended. Choose this only if you need to use programs that are not certified for Windows 7 because they do not support User Account Control.

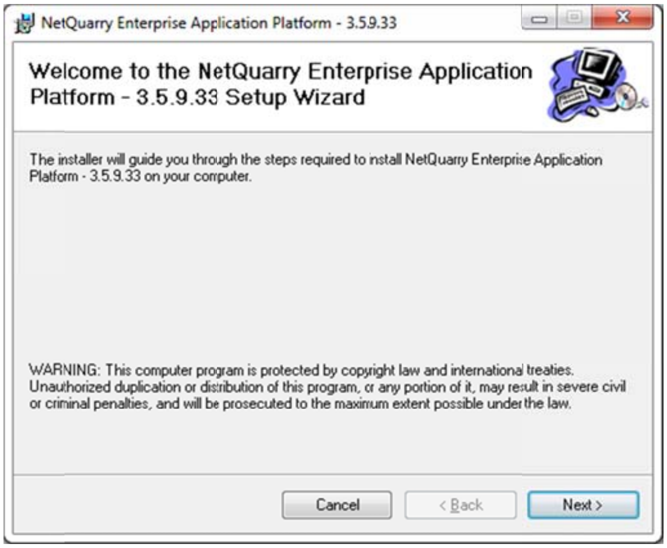
If you do not make this change, then you must perform the platform installation process via an "Administrator" command prompt.

NetQuarry Platform Installation

You should install the NetQuarry Platform after the prerequisites. NetQuarry consists of a single MSI (Windows Installer) file. You may install this file by running it from the command line or Explorer. During installation, environment variables for NQROOT and NQBIN are created. These point to %SYSTEM_ROOT%\InetPub\wwwroot\<virtual folder\ and %SYSTEM_ROOT%\Inetpub\wwwroot\<virtual folder>\bin.

You can download the latest NetQuarry Installer from https://dev.netquarry.com/install/lastbuild/ You will be prompted for credentials. User: nqinstaller, pwd: netquarry1470.

Create a folder called C:\NetQuarry\Customers\IssueTrak and download the installer to that folder. Once downloaded you can simply launch the install process of the NetQuarry Platform which will only take a few seconds to complete.



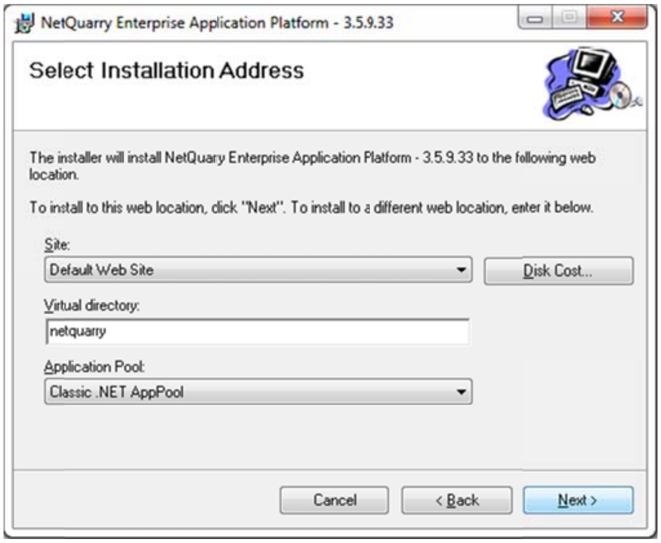
NetQuarry Enterprise Application Platform - 3.5.9.33

Welcome to the NetQuarry Enterprise Application Platform - 3.5.9.33 Setup Wizard

The installer will guide you through the steps required to install NetQuarry Enterprise Application Platform - 3.5.9.33 on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

You will be prompted to choose an Application Pool with which to associate the installation (here you choose the application pool with the "classic" managed pipeline mode). Also you can choose whether to install platform into a virtual directory. If you wish to install the platform to the root of IIS, then don't provide a virtual folder name. Here we will be installing the platform to the **netquarry** virtual folder.



Select Installation Address

The installer will install NetQuary Enterprise Application Platform - 3.5.9.33 to the following web location.

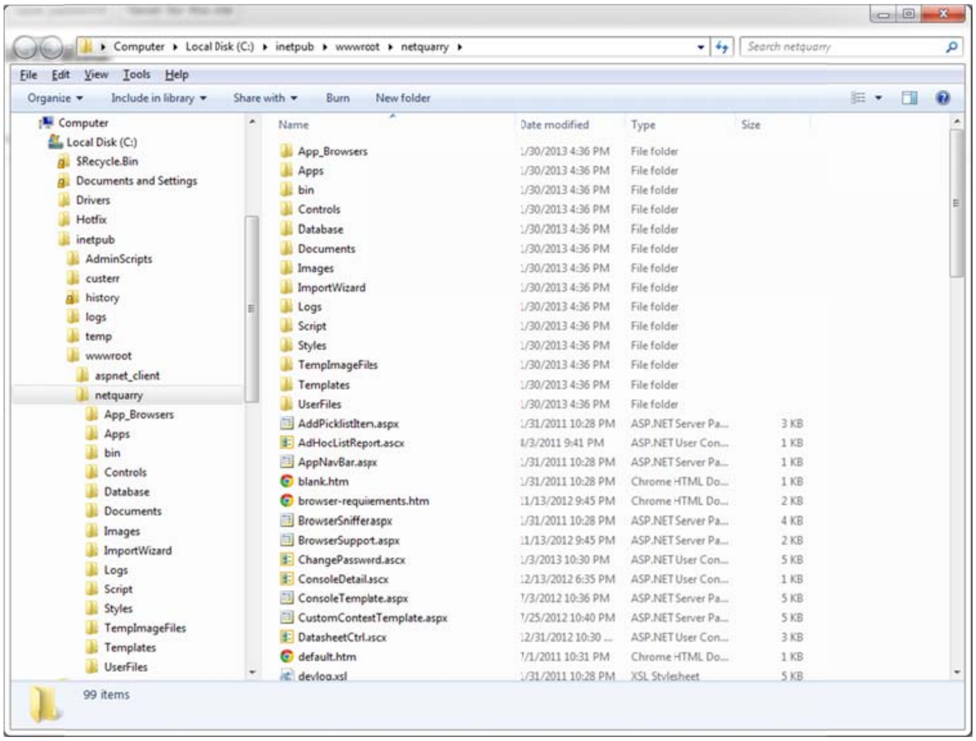
To install to this web location, click "Next". To install to a different web location, enter it below.

Site: Default Web Site

Virtual directory: netquarry

Application Pool: Classic .NET AppPool

Once the installation is complete, you can confirm the platform is installed, by navigating to the installation folder, %NQROOT%. If you chose to follow the instructions it will be C:\inetpub\wwwroot\netquarry



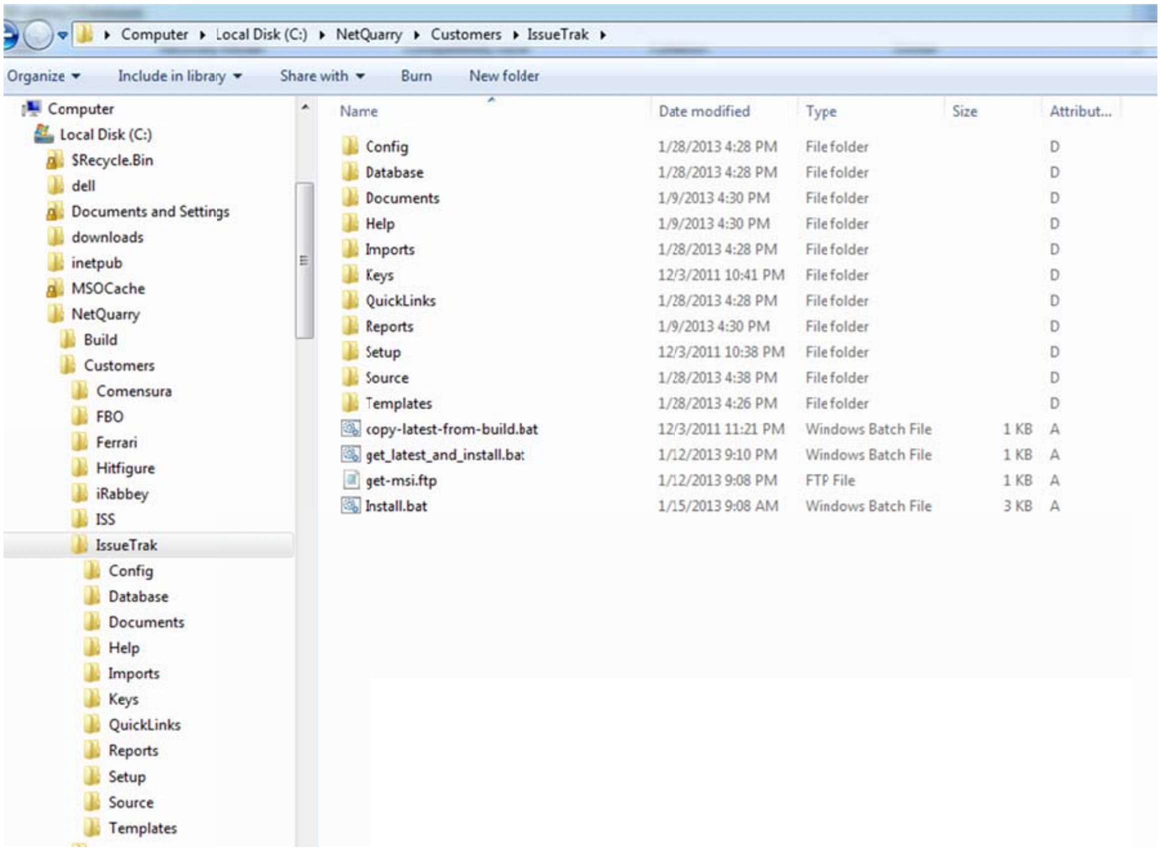
Name	Date modified	Type	Size
App_Browsers	1/30/2013 4:36 PM	File folder	
Apps	1/30/2013 4:36 PM	File folder	
bin	1/30/2013 4:36 PM	File folder	
Controls	1/30/2013 4:36 PM	File folder	
Database	1/30/2013 4:36 PM	File folder	
Documents	1/30/2013 4:36 PM	File folder	
Images	1/30/2013 4:36 PM	File folder	
ImportWizard	1/30/2013 4:36 PM	File folder	
Logs	1/30/2013 4:36 PM	File folder	
Script	1/30/2013 4:36 PM	File folder	
Styles	1/30/2013 4:36 PM	File folder	
TempImageFiles	1/30/2013 4:36 PM	File folder	
Templates	1/30/2013 4:36 PM	File folder	
UserFiles	1/30/2013 4:36 PM	File folder	
AddPicklistItem.aspx	1/31/2011 10:28 PM	ASP.NET Server Pa...	3 KB
AdHocListReport.ascx	8/3/2011 9:41 PM	ASP.NET User Con...	1 KB
AppNavBar.aspx	1/31/2011 10:28 PM	ASP.NET Server Pa...	1 KB
blank.htm	1/31/2011 10:28 PM	Chrome HTML Do...	1 KB
browser-requiements.htm	11/13/2012 9:45 PM	Chrome HTML Do...	2 KB
BrowserSnifferaspx	1/31/2011 10:28 PM	ASP.NET Server Pa...	4 KB
BrowserSupport.aspx	11/13/2012 9:45 PM	ASP.NET Server Pa...	2 KB
ChangePassword.ascx	1/3/2013 10:30 PM	ASP.NET User Con...	5 KB
ConsoleDetail.ascx	12/13/2012 6:35 PM	ASP.NET User Con...	1 KB
ConsoleTemplate.aspx	7/3/2012 10:36 PM	ASP.NET Server Pa...	5 KB
CustomContextTemplate.aspx	7/25/2012 10:40 PM	ASP.NET Server Pa...	5 KB
DatasheetCtrl.ascx	12/31/2012 10:30 ...	ASP.NET User Con...	3 KB
default.htm	7/1/2011 10:31 PM	Chrome HTML Do...	1 KB
devlog.xsl	1/31/2011 10:28 PM	XSL Stylesheet	5 KB

Also confirm that the environment variable %NQBIN% is mapped to %NQROOT%\bin.

Setting up IssueTrak Environment

On your C: drive, create a folder structure for C\NetQuarry\Customers. Then navigate to %NQROOT%\Documents\Tutorials. Copy the IssueTrak subfolder and all of its contents to the C:\NetQuarry\Customers folder.

Once the copy is completed, you should see a folder structure similar to this…



Name	Date modified	Type	Size	Attribut...
Config	1/28/2013 4:28 PM	File folder		D
Database	1/28/2013 4:28 PM	File folder		D
Documents	1/9/2013 4:30 PM	File folder		D
Help	1/9/2013 4:30 PM	File folder		D
Imports	1/28/2013 4:28 PM	File folder		D
Keys	12/3/2011 10:41 PM	File folder		D
QuickLinks	1/28/2013 4:28 PM	File folder		D
Reports	1/9/2013 4:30 PM	File folder		D
Setup	12/3/2011 10:38 PM	File folder		D
Source	1/28/2013 4:38 PM	File folder		D
Templates	1/28/2013 4:26 PM	File folder		D
copy-latest-from-build.bat	12/3/2011 11:21 PM	Windows Batch File	1 KB	A
get_latest_and_install.bat	1/12/2013 9:10 PM	Windows Batch File	1 KB	A
get-msi.ftp	1/12/2013 9:08 PM	FTP File	1 KB	A
Install.bat	1/15/2013 9:08 AM	Windows Batch File	3 KB	A

We want you to copy these files from the installation folder to another folder because each time you install the NetQuarry Platform, the content of the installed folders is deleted and recreated. Copying this content to new folder ensures any updates to the platform do not affect any of the work/changes you make to the IssueTrak application.

Also note the location we copy the files to, is referenced in various automation batch files that are described in the next section. If you deviate from the described steps, you will need to make some changes to the automation batch files.

Automating Some Tasks

The platform installation comes with a set of convenience tools, shortcuts and batch files to automate some of the repetitive, or infrequently used tasks. Such as downloading and installing a new NetQuarry Platform. Scripting database changes and loading and saving metadata files.

These shortcuts are found in the C:\NetQuarry\Customers\IssueTrak\NQLinks folder. This folder has three subfolders. Code, Database, Tools.

In the Tools folder there is a self extracting executable that you should run to unpack the set of required tools shortcuts for this tutorial. Once you have run the self extracting executable, you can delete it.

The content of these folders can be added as a new toolbar to your TaskBar (right click on TaskBar, choose "Toolbars" and "New Toolbar". Then navigate to the NQLinks folder. Choose the NQLinks folder or drill down and select each of the sub folders individually.

Code NQLinks

“Build IssueTrak” provides a shortcut to a batch file that compiles all the code associated with the IssueTrak application.

The GenCode folder provides shortcuts to generate class objects from the meta data.

Database NQLinks

“Update IssueTrakDB” provides a shortcut to a set of scripts that update the schema and meta data for the Issuetrak application.

“Update NQ DB” provides a shortcut to a set of scripts that update the NetQuarry studio schema and metadata.

NQ Apps NQLinks

“NQ Log Viewer” provides a shortcut to a program that displays debug output information from the NetQuarry platform (web applications, scheduled tasks, studio).

“NQ Studio” provides a shortcut to launch the NetQuarry Studio application.

“NQ Task Runner” provides a shortcut to a tool that executes a scheduled task for any configured application to assist in debugging potential problems.

Tools NQLinks

“Get Latest Platform and Install” provides a shortcut to a set of scripts that download the latest NetQuarry platform, uninstalls the current platform, reinstalls the latest platform and runs the necessary scripts to update the NetQuarry studio schema and metadata.

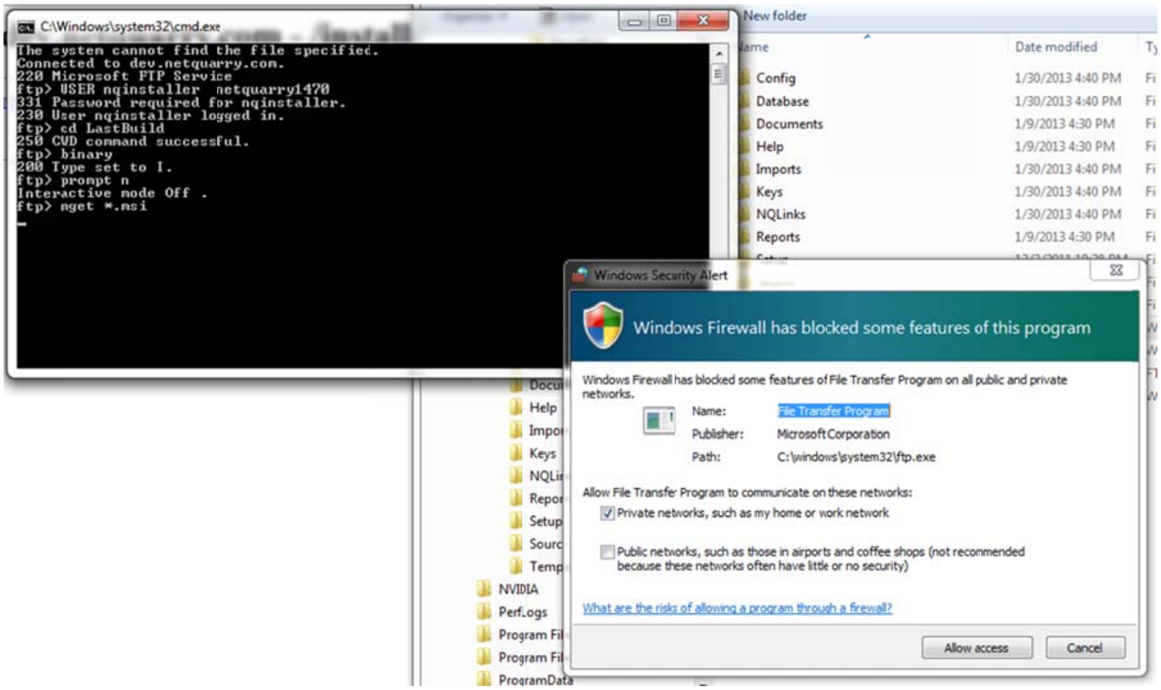
Reset App Pool.bat provides a way to quickly (much more quickly than resetting IIS) recycle the app pool during development so you can release any custom extensions from memory during development.

Customizing/ Running Platform Install Scripts

The next step is to confirm you can automate the platform installation process.

Navigate to the C:\NetQuarry\Customers\Issuetrak folder.

Execute the “copy-latest-from-build.bat” script to confirm that this downloads the latest version of the NetQuarry Platform installer to the local folder. You may be prompted by the windows firewall to allow the windows ftp client to access the internet.



```
The system cannot find the file specified.
Connected to dev.netquarry.com.
220 Microsoft FTP Service
ftp> USER nqinstaller  netquarry1470
331 Password required for nqinstaller.
230 User nqinstaller logged in.
ftp> cd LastBuild
250 CWD command successful.
ftp> binary
200 Type set to I.
ftp> prompt n
Interactive mode Off .
ftp> mget *.msi
_
```

Windows Firewall has blocked some features of this program

Windows Firewall has blocked some features of File Transfer Program on all public and private networks.

Name: File Transfer Program
Publisher: Microsoft Corporation
Path: C:\windows\system32\ftp.exe

Allow File Transfer Program to communicate on these networks:

☑ Private networks, such as my home or work network

□ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing a program through a firewall?
Allow access
Cancel

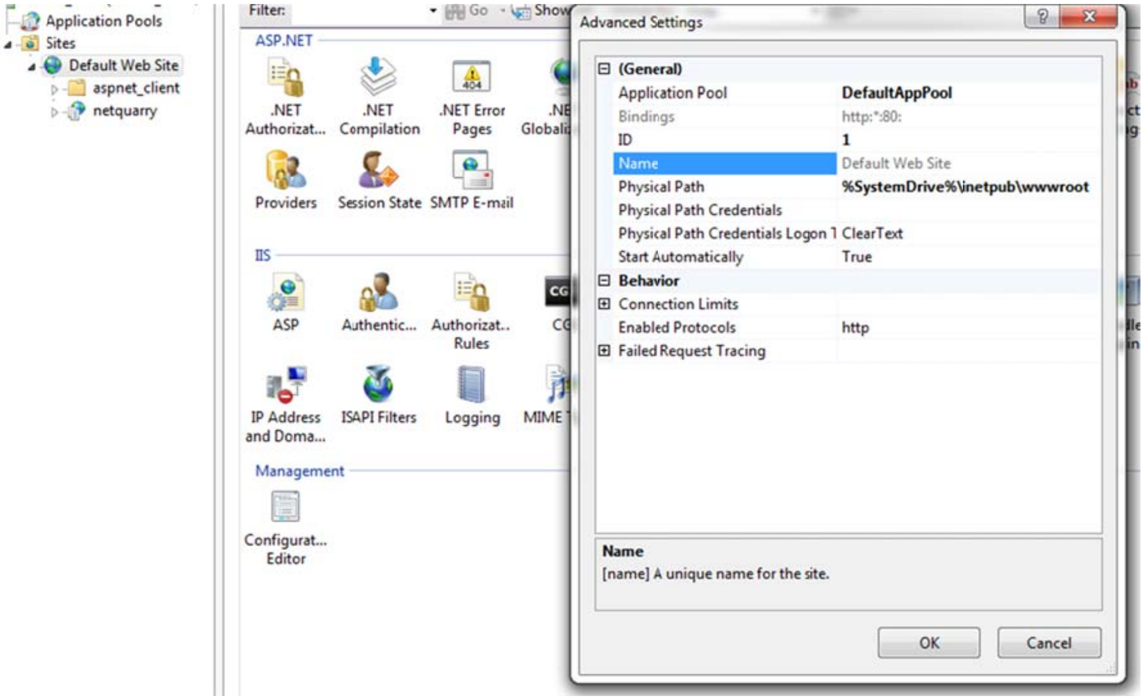
In the same folder, C:\NetQuarry\Customers\IssueTrak, edit the Install.bat script.

```
@echo off

@echo Un-installing NetQuarry Platform...
call "%SystemRoot%\System32\MsiExec.exe" /x {642ADDFC-6BBC-4C4B-A8C9-377727C18024} /passive

@echo Installing NetQuarry Platform...
call "%SystemRoot%\System32\msiexec.exe" /i "NetQuarry.msi" /log "C:\NetQuarry\Customers\Issuetrak\install-
platform.log" INSTALLDIR="C:\inetpub\wwwroot" TARGETAPPPOOL="Classic .NET AppPool" TARGETSITE="/LM/W3SVC/1"
TARGETVDIR="" /passive

IF NOT EXIST %NQBIN%\eap.core.dll GOTO ERROREXIT

@echo Installing core databases...
cd %NQROOT%\Database\Scripts

@echo Install studio repository...
call %NQROOT%\Database\Scripts\db-install.bat

@echo ----------------------------------------------------------------------------------
@echo Updating local .config files
@echo ----------------------------------------------------------------------------------
IF EXIST C:\NetQuarry\Customers\IssueTrak\Config\web.config copy C:\NetQuarry\Customers\IssueTrak\Config\web.config
%NQROOT%\web.config
IF EXIST C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.Tools.exe.config copy
C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.Tools.exe.config %NQBIN%\EAP.Tools.exe.config
IF EXIST C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.Tools.CodeGen.exe.config copy
C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.Tools.CodeGen.exe.config %NQBIN%\EAP.Tools.CodeGen.exe.config
IF EXIST C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.Tools.TaskRunner.exe.config copy
C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.Tools.TaskRunner.exe.config %NQBIN%\EAP.Tools.TaskRunner.exe.config
IF EXIST C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.Scheduler.exe.config copy
C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.Scheduler.exe.config %NQBIN%\EAP.Scheduler.exe.config
IF EXIST C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.VSAddIn.dll.config copy
C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.VSAddIn.dll.config %NQBIN%\EAP.VSAddIn.dll.config
IF EXIST C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.Reporting.config copy
C:\NetQuarry\Customers\IssueTrak\Config\bin\EAP.Reporting.config %NQBIN%\EAP.Reporting.config

@echo ----------------------------------------------------------------------------------
@echo restore customized style and js
@echo ----------------------------------------------------------------------------------
IF EXIST C:\NetQuarry\Customers\IssueTrak\Config\web\Styles\custom.css copy
C:\NetQuarry\Customers\IssueTrak\Config\web\Styles\custom.css %NQROOT%\Apps\IssueTrak\Styles\custom.css
IF EXIST C:\NetQuarry\Customers\IssueTrak\Config\web\Script\IssueTrak.js copy
C:\NetQuarry\Customers\IssueTrak\Config\web\Script\IssueTrak.js %NQROOT%\Apps\IssueTrak\Script\IssueTrak.js

@echo
@echo Updating assembly path
SQLCMD -S %SERVER% -Q "USE [issues_meta]; UPDATE issues_meta..xmt_components SET assembly_path = '%%NQBIN%%' WHERE
module_key='core_components' AND assembly_path IS NOT NULL;"

goto END

:ERROREXIT
@echo One or more errors occurred installing or the installation was cancelled.
GOTO FINAL

:END
@echo Install complete...
pause

:FINAL
```

In the batch file. DO NOT remove or modify the first command in script (gray color). Before a new platform can be installed, it must first be uninstalled.

The highlighted command section, should be modified to suit your environment. The TARGETAPPPOOL should be changed to the app pool you selected in the initial installation.

The TARGETSITE should be modified to match the site for the virtual folder. To determine the TARGETSITE, open the IIS Manager and identify the Parent Site of your virtual folder. Then, select that Site and click on Advanced Settings.



⊟ **(General)**	
Application Pool	**DefaultAppPool**
Bindings	http:*:80:
ID	**1**
Name	Default Web Site
Physical Path	**%SystemDrive%\inetpub\wwwroot**
Physical Path Credentials	
Physical Path Credentials Logon T	ClearText
Start Automatically	True
⊟ **Behavior**	
⊞ Connection Limits	
Enabled Protocols	http
⊞ Failed Request Tracing	

Name
[name] A unique name for the site.

The ID property value (1, 2, etc) should be reflected in the site's ID in the batch file ("/LM/W3SVC/1", or "/LM/W3SVC/2" respectively).

The TARGETVDIR denotes the virtual folder to install the application to. If you chose not to install the application to a virtual folder during the initial setup, then you should remove the TARGETVDIR option from the install script. Or set TARGETVDIR=""

Save those changes and run the Install.bat script. This will uninstall the NetQuarry Platform and Reinstall in exactly the same place with all the same original settings.

Now you can run the get_latest_and_install.bat script that wraps the execution of the latest platform download and installation.

Running Database Creation/Update Scripts

The next step is to create the database object and schema. There are four databases that need to be created.

- issues_data – which contains the operational data to store the issue trak information.

- issues_meta – which contains the metadata for the IssueTrak application.

- issues_doc – which is the storage database for uploaded documents.

- issues_log – which is the storage database for logging functions.

The issues_doc and issues_log databases are not required for the IssueTrak to work. In the absence of these extra databases, document storage and logging storage could be held in the issues_data database. The splitting of database usage is a convenience for managing both the size, performance and portability of applications.

It's possible to create and populate the database objects manually, but we have provided a shortcut to a script that performs all the necessary tasks. Which are…

- Create database if they do not exist.
- Set the appropriate role member permissions.
- Populate the databases with the necessary schema.
- Populate the issues_meta database with the necessary metadata.
- Populate (seed) the issues_data database with any necessary data.

Navigate to the C:\NetQuarry\Customers\IssueTrak\NQLinks\Database folder and run the Update Issue Trak DB script shortcut. If you have added the NQLinks as a toolbar on the TaskBar, access it via that shortcut.

The script utilizes/execute various SQL and meta files depending on the required purpose. The following table shows where these files are located.

Application	**File Type**	**File Location**
NQ Platform (NQ)	.SQL	%NQROOT%\Database\Scripts
NQ Platform (NQ)	.meta	%NQROOT%\Database\Scripts\Metadata
IssueTrak (IT)	.SQL	C:\NetQuarry\Customers\IssueTrak\Database
IssueTrak (IT)	.meta	C:\NetQuarry\Customers\IssueTrak\Database\Metadata

Upon execution of the database script, your output should look something like this.

Content	**Notes**

`*************************************************************`
`**** Creating IssueTrak Database objects ****`
`*************************************************************`
	(IT) issues_database_create.sql Creates issues_data, issues_doc_issues_log and issues_meta databases

`*************************************************************`
`**** UPDATING issues_meta SYSOP SCHEMA ****`
`*************************************************************`
`Current xot_schema_version [nq-op-schema] = 0. Required version: 4`
`Adding xot_api_request_log`
`Adding xot_api_request_log.[thread_id] int NULL`
`Adding xot_api_request_log.[device_type] varchar(100) NULL`
`Adding xot_api_request_log.[device_os] varchar(100) NULL`
`Adding xot_api_request_log.[device_os_version] varchar(100) NULL`
`Adding xot_api_request_log.[machine_nm] [varchar] (100) NULL`
`Adding xot_api_request_log.ip_address`
`Adding xot_api_request_log.app_version varchar(32)`
`Adding xot_api_request_log.related_id, related_guid, related_int...`
`Adding xot_api_request_log.request_duration`
	(NQ) nq_op_schema.sql – contains the scripts to create platform specific operational schema in the meta database

Content

```
Adding xot_api_request_log.ack_guid (indexed), is_acked
Adding xot_service_tokens.callback_endpoint_url varchar(1000)
Adding xot_device_token.push_token varchar(500)
Adding xot_device_token.device_os varchar(200)
Adding xot_device_token.app_version varchar(32)
Adding xot_device_token.[device_type] varchar(100) NULL
Adding xot_device_token.[device_os_version] varchar(100) NULL
Adding xot_device_token.updated_dt
Adding xot_saved_filters.usage...
Adding xot_saved_filters.page_element_id...
Altering xot_saved_filters.module_key to 50 chars
Adding xot_saved_filters.flt_desc_verbose.
CREATING xot_account_log...
add xot_account_log.error_message...
add xot_account_log.action_taken_by_id...
CREATING xot_request_log_rollup...
CREATING xot_request_log_err_rollup...
CREATING xot_tenant_permissions...
Creating xot_view_state table
Creating xot_imports table
Creating xot_adhoc_list_reports table
add xot_bookmarks.visits, updated_dt...
CREATING xot_user_permissions...
Increased xot_browser_history.ip_address 45 to cover ipv6 addresses (with
ipv4 translations)
Adding xot_browser_history.eap_utkn.
Adding xot_failed_sql_log.data_source_key, attr_bits...
Adding xot_alerts.delivery_type_id, auto_close_secs, css_class...

*************************************************************
*** UPDATING issues_meta SYSTEM SCHEMA ****
*************************************************************
DELETING DATA FROM TABLE xmt_locales
DELETING DATA FROM TABLE xmt_obj_type_rel
DELETING DATA FROM TABLE xmt_obj_types
DELETING DATA FROM TABLE xmt_properties
DELETING DATA FROM TABLE xmt_text
INSERTING DATA INTO TABLE xmt_locales
INSERTING DATA INTO TABLE xmt_obj_type_rel
INSERTING DATA INTO TABLE xmt_obj_types
INSERTING DATA INTO TABLE xmt_properties
INSERTING DATA INTO TABLE xmt_text

*************************************************************
**** UPDATING METADATA IN issues_meta ****
*************************************************************
Loading issues metadata...
----------------------------------------------------------------------
Script Path: C:\inetpub\wwwroot\Database\Scripts
Metadata Path: C:\inetpub\wwwroot\Database\Scripts\Metadata
MetaUtil.exe: C:\inetpub\wwwroot\bin"\EAP.Tools.MetaUtil.EXE"
Connection String: "Provider=SQLOLEDB;Data Source=.;Initial
Catalog=issues_meta;Integrated Security=SSPI"
----------------------------------------------------------------------
Updating core schema for issues_meta
Current xot_schema_version [nq-op-schema] = 4. Required version: 4
Msg 50000, Level 16, State 1, Server ISSUETRAK_VM, Line 6
 *** This version of nq-op-schema.sql has already been executed. ***
Updating core schema for issues_data
Current xot_schema_version [nq-op-schema] = 0. Required version: 4
Adding xot_api_request_log
Adding xot_api_request_log.[thread_id] int NULL
Adding xot_api_request_log.[device_type] varchar(100) NULL
Adding xot_api_request_log.[device_os] varchar(100) NULL
Adding xot_api_request_log.[device_os_version] varchar(100) NULL
Adding xot_api_request_log.[machine_nm] [varchar] (100) NULL
Adding xot_api_request_log.ip_address
Adding xot_api_request_log.app_version varchar(32)
Adding xot_api_request_log.related_id, related_guid, related_int...
Adding xot_api_request_log.request_duration
```

Notes

(NQ) meta-schema.sql – contains the scripts to create the plaltform schema that is required for a database to act as the metadata repository.

(NQ) nq_op_schema.sql – contains the scripts to create platform specific operational schema in the meta database

Content

```
Adding xot_api_request_log.ack_guid (indexed), is_acked
Adding xot_service_tokens.callback_endpoint_url varchar(1000)
Adding xot_device_token.push_token varchar(500)
Adding xot_device_token.device_os varchar(200)
Adding xot_device_token.app_version varchar(32)
Adding xot_device_token.[device_type] varchar(100) NULL
Adding xot_device_token.[device_os_version] varchar(100) NULL
Adding xot_device_token.updated_dt
Adding xot_saved_filters.usage...
Adding xot_saved_filters.page_element_id...
Altering xot_saved_filters.module_key to 50 chars
Adding xot_saved_filters.flt_desc_verbose.
CREATING xot_account_log...
add xot_account_log.error_message...
add xot_account_log.action_taken_by_id...
CREATING xot_request_log_rollup...
CREATING xot_request_log_err_rollup...
CREATING xot_tenant_permissions...
Creating xot_view_state table
Creating xot_imports table
Creating xot_adhoc_list_reports table
add xot_bookmarks.visits, updated_dt...
CREATING xot_user_permissions...
Increased xot_browser_history.ip_address 45 to cover ipv6 addresses (with
ipv4 translations)
Adding xot_browser_history.eap_utkn.
Adding xot_failed_sql_log.data_source_key, attr_bits...
Adding xot_alerts.delivery_type_id, auto_close_secs, css_class...
----------------------------------------------------------------------
Loading application: issues
----------------------------------------------------------------------
EAP.MetaUtil.exe C:\inetpub\wwwroot\bin"\EAP.Tools.MetaUtil.EXE"  -load
C:\NetQuarry\Customers\IssueTrak\Database\Metadata\ "Provider=SQLOLEDB;Data
Source=.;Initial Catalog=issues_meta;Integrated Security=SSPI" issuetrak
Loading application metadata: issuetrak
Application metadata load failed. File does not exist:
C:\NetQuarry\Customers\IssueTrak\Database\Metadata\issuetrak-
application.meta
----------------------------------------------------------------------
```

Notes

Loading the IssueTrak meta data for the ‘issuestrak’ application into the issues_meta database. Since this is the first time the script has been run. There is no ‘IssueTrak’ application and therefore the script fails at this point.

```
Loading application: issuespr
----------------------------------------------------------------------
EAP.MetaUtil.exe C:\inetpub\wwwroot\bin"\EAP.Tools.MetaUtil.EXE"  -load
C:\NetQuarry\Customers\IssueTrak\Database\Metadata\ "Provider=SQLOLEDB;Data
Source=.;Initial Catalog=issues_meta;Integrated Security=SSPI" issuetrakpr
Loading application metadata: issuetrakpr
Module loaded: IssueTrakpr
Loading modules for application: issuetrakpr
Start: 11/2/2018 3:05:19 PM, End: 11/2/2018 3:05:19 PM. Elapsed: 0.046875
seconds.
Application 'issuetrakpr' successfully loaded.
----------------------------------------------------------------------
```

Loading the IssueTrak meta data for the ‘issuetrakpr’ application into the issues_meta database. Since this is the first time the script has been run. There is no ‘issuetrakpr’ application and therefore the script fails at this point.

```
Loading application: system
----------------------------------------------------------------------
EAP.MetaUtil.exe C:\inetpub\wwwroot\bin"\EAP.Tools.MetaUtil.EXE"  -load
C:\inetpub\wwwroot\Database\Scripts\Metadata "Provider=SQLOLEDB;Data
Source=.;Initial Catalog=issues_meta;Integrated Security=SSPI" system
Loading application metadata: system
Module loaded: system
Loading modules for application: system
Module loaded: activity_log
Module loaded: audit
Module loaded: core_components
Module loaded: core_navigators
Module loaded: developer_tools
Module loaded: eap_export
Module loaded: eap_package
Module loaded: filters
Module loaded: import
Module loaded: password
Module loaded: password_controls
```

Loading the NQ platform metadata into the issues_meta database.

Content

```
Module loaded: postit
Module loaded: reporting
Module loaded: system
Module loaded: tenanting
Module loaded: user_preference
Module loaded: vocab
BulkCopy: xmt_modules. Rows: 17
BulkCopy: xmt_datamappers. Rows: 49
BulkCopy: xmt_fields. Rows: 581
BulkCopy: xmt_text. Rows: 2635
BulkCopy: xmt_properties. Rows: 1852
BulkCopy: xmt_extensions. Rows: 37
BulkCopy: xmt_pages. Rows: 55
BulkCopy: xmt_page_elements. Rows: 65
BulkCopy: xmt_picklists. Rows: 52
BulkCopy: xmt_picklist_items. Rows: 109
BulkCopy: xmt_components. Rows: 154
BulkCopy: xmt_parameters. Rows: 1682
BulkCopy: xmt_navigators. Rows: 10
BulkCopy: xmt_nav_targets. Rows: 35
BulkCopy: xmt_scheduled_tasks. Rows: 2
BulkCopy: xmt_filters. Rows: 9
BulkCopy: xmt_named_filters. Rows: 2
BulkCopy: xmt_obj_profile_permissions. Rows: 52
BulkCopy: xmt_flavor_names. Rows: 1
BulkCopy: xmt_param_types. Rows: 219
BulkCopy: xmt_text_types. Rows: 42
BulkCopy: xmt_data_types. Rows: 169
BulkCopy: xmt_versions. Rows: 3
BulkCopy: xmt_text_type_object_type. Rows: 117
Start: 11/2/2018 3:05:20 PM, End: 11/2/2018 3:05:21 PM. Elapsed: 1.203125
seconds.
Application 'system' successfully loaded.
----------------------------------------------------------------------
Metdata load complete
----------------------------------------------------------------------
*************************************************************
**** UPDATING issues_data SYSOP SCHEMA ****
*************************************************************
Current xot_schema_version [nq-op-schema] = 4. Required version: 4
Msg 50000, Level 16, State 1, Server ISSUETRAK_VM, Line 6
 *** This version of nq-op-schema.sql has already been executed. ***
*************************************************************
**** UPDATING issues_logs SYSOP SCHEMA ****
*************************************************************
Current xot_schema_version [nq-op-schema] = 0. Required version: 4
Adding xot_api_request_log
Adding xot_api_request_log.[thread_id] int NULL
Adding xot_api_request_log.[device_type] varchar(100) NULL
Adding xot_api_request_log.[device_os] varchar(100) NULL
Adding xot_api_request_log.[device_os_version] varchar(100) NULL
Adding xot_api_request_log.[machine_nm] [varchar] (100) NULL
Adding xot_api_request_log.ip_address
Adding xot_api_request_log.app_version varchar(32)
Adding xot_api_request_log.related_id, related_guid, related_int...
Adding xot_api_request_log.request_duration
Adding xot_api_request_log.ack_guid (indexed), is_acked
Adding xot_service_tokens.callback_endpoint_url varchar(1000)
Adding xot_device_token.push_token varchar(500)
Adding xot_device_token.device_os varchar(200)
Adding xot_device_token.app_version varchar(32)
Adding xot_device_token.[device_type] varchar(100) NULL
Adding xot_device_token.[device_os_version] varchar(100) NULL
Adding xot_device_token.updated_dt
Adding xot_saved_filters.usage...
Adding xot_saved_filters.page_element_id...
Altering xot_saved_filters.module_key to 50 chars
Adding xot_saved_filters.flt_desc_verbose.
CREATING xot_account_log...
add xot_account_log.error_message...
```

Notes

(NQ) nq_op_schema.sql – for issues_data database

(NQ) nq_op_schema.sql – for issues_log database

Content

```
add xot_account_log.action_taken_by_id...
CREATING xot_request_log_rollup...
CREATING xot_request_log_err_rollup...
CREATING xot_tenant_permissions...
Creating xot_view_state table
Creating xot_imports table
Creating xot_adhoc_list_reports table
add xot_bookmarks.visits, updated_dt...
CREATING xot_user_permissions...
Increased xot_browser_history.ip_address 45 to cover ipv6 addresses (with
ipv4 translations)
Adding xot_browser_history.eap_utkn.
Adding xot_failed_sql_log.data_source_key, attr_bits...
Adding xot_alerts.delivery_type_id, auto_close_secs, css_class...

*************************************************************
***** UPDATING issues_data OP SCHEMA ******
*************************************************************

*************************************************************
***** UPDATING schema                                  ******
*************************************************************

*************************************************************
***** UPDATING indexes                                 ******
*************************************************************

*************************************************************
***** UPDATING views                                   ******
*************************************************************

*************************************************************
***** UPDATING procs                                   ******
*************************************************************

*************************************************************
***** UPDATING operational data                        ******
*************************************************************
CREATE TABLE [dbo].[schema_upgrade_versioning]
Performing update for SEED_COMPANY
(1 row affected)
Performing update for SEED_DOC_TYPES
(3 rows affected)
Performing update for SEED_ADMIN_USER
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
Performing update for SEED_PROJECT
(1 row affected)
(9 rows affected)
(5 rows affected)
(1 row affected)
(8 rows affected)
(3 rows affected)
(11 rows affected)
Press any key to continue . . .
```

Notes

(IT) issues_data-alter-schema.sql – contains schema scripts for creating operational tables

(IT) issues_data-indexes.sql – contains scripts for creating indexes on operational tables

(IT) issues_data-views.sql – contains scripts for creating views on operational tables

(IT) issues_data-procs.sql – contains scripts for creating stored procedures on operational tables

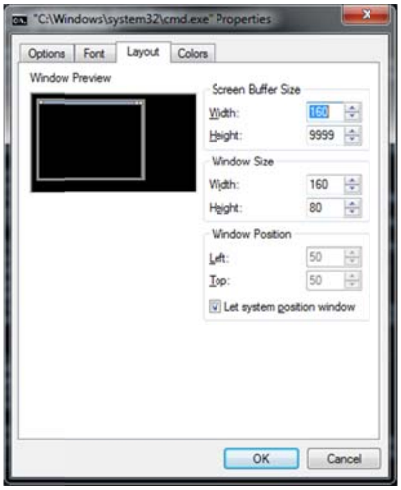
(IT) issues_data-op-updates.sql – contains scripts for seeding data into operational tables

Script has a pause for you to review the results of the database load

Note that the script is re-entrant and destructive. If you have metadata in the issues_meta database it will be lost upon running the script.

As your project grows, the length of the output will also grow. It is recommended that you increase the storage buffer of your command prompt window. On Title bar of command window, right click and choose properties.

On the layout tab, make sure your Screen Buffer Size height is set to the maximum 9999



Screen Buffer Size
Width: 160
Height: 9999

Window Size
Width: 160
Height: 80

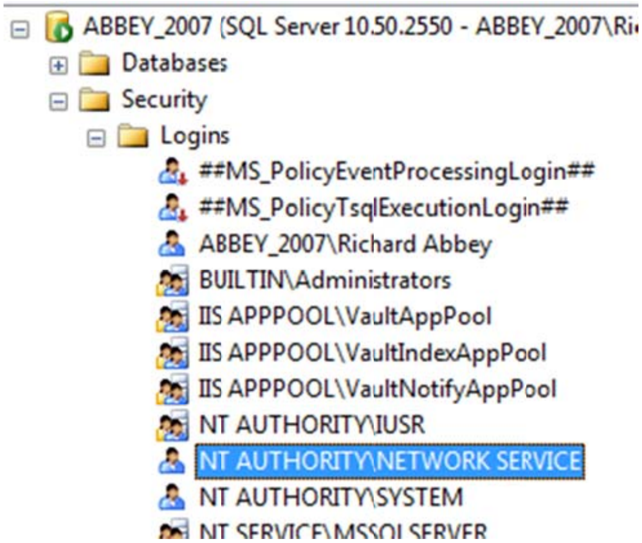
Window Position
Left: 50
Top: 50

☑ Let system position window

Setting Database Permissions

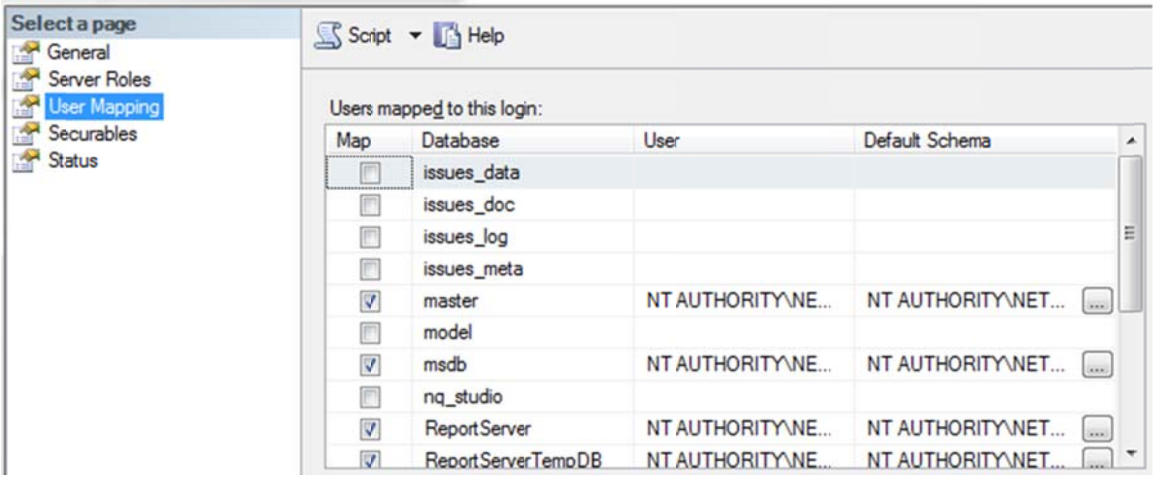
To ensure you can connect to the databases you just created, we need to set the appropriate database permissions. Here, we are going to assume that you are an administrator on your computer.

Open SQL Studio and Expand the Security and Logins. Select the NT AUTHORITY\NETWORK SERVICE user, or the user you have chosen as the identity of your app pool.



ABBEY_2007 (SQL Server 10.50.2550 - ABBEY_2007\Ri
Databases
Security
Logins
##MS_PolicyEventProcessingLogin##
##MS_PolicyTsqlExecutionLogin##
ABBEY_2007\Richard Abbey
BUILTIN\Administrators
IIS APPPOOL\VaultAppPool
IIS APPPOOL\VaultIndexAppPool
IIS APPPOOL\VaultNotifyAppPool
NT AUTHORITY\IUSR
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
NT SERVICE\MSSQLSERVER

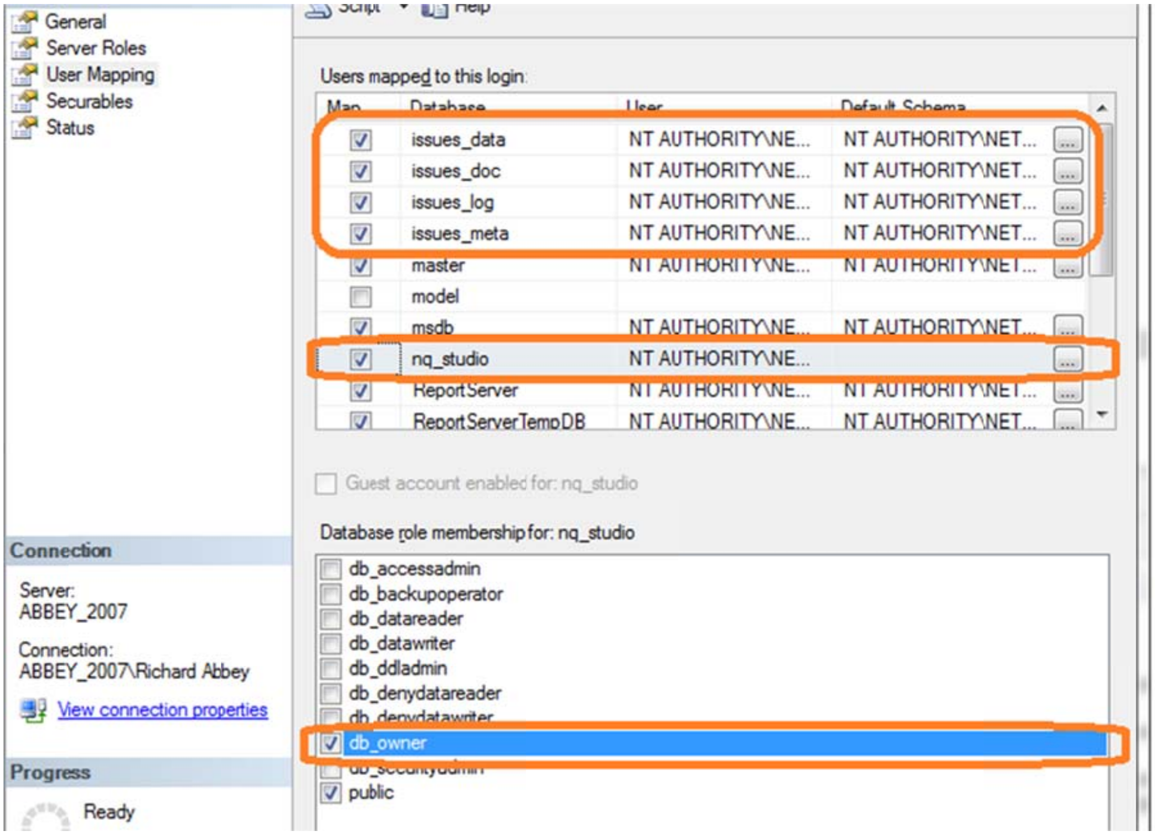
Right click on this Login and choose Properties. Select the User Mappings option.



Users mapped to this login:

Map	Database	User	Default Schema
☐	issues_data		
☐	issues_doc		
☐	issues_log		
☐	issues_meta		
☑	master	NT AUTHORITY\NE...	NT AUTHORITY\NET...
☐	model		
☑	msdb	NT AUTHORITY\NE...	NT AUTHORITY\NET...
☐	nq_studio		
☑	ReportServer	NT AUTHORITY\NE...	NT AUTHORITY\NET...
☑	ReportServerTempDB	NT AUTHORITY\NE...	NT AUTHORITY\NET...

For each of the highlighted databases, ensure the membership role is set to db_owner



Users mapped to this login:

Map	Database	User	Default Schema
☑	issues_data	NT AUTHORITY\NE...	NT AUTHORITY\NET...
☑	issues_doc	NT AUTHORITY\NE...	NT AUTHORITY\NET...
☑	issues_log	NT AUTHORITY\NE...	NT AUTHORITY\NET...
☑	issues_meta	NT AUTHORITY\NE...	NT AUTHORITY\NET...
☑	master	NT AUTHORITY\NE...	NT AUTHORITY\NET...
☐	model		
☑	msdb	NT AUTHORITY\NE...	NT AUTHORITY\NET...
☑	nq_studio	NT AUTHORITY\NE...	
☑	ReportServer	NT AUTHORITY\NE...	NT AUTHORITY\NET...
☑	ReportServerTempDB	NT AUTHORITY\NE...	NT AUTHORITY\NET...

☐ Guest account enabled for: nq_studio

Database role membership for: nq_studio

☐ db_accessadmin
☐ db_backupoperator
☐ db_datareader
☐ db_datawriter
☐ db_ddladmin
☐ db_denydatareader
☐ db_denydatawriter
☑ db_owner
☐ db_securityadmin
☑ public

Server:
ABBEY_2007

Connection:
ABBEY_2007\Richard Abbey

Ready

Click OK. Repeat this same process for the Login, BUILTIN\Administrators

Setting Folder Permissions

The last step to perform before continuing with the creation of the IssueTrak application is to ensure that various folders exist and have the necessary access permissions.

For all folders quoted below. Ensure that both your current logon credentials AND the user specified as the application pool Identity (e.g. NT AUTHORITY\NETWORK SERVICE) has full access permissions. If the folder does not exist. Create the folder and set the appropriate permissions. If the folder is empty, then create an empty text file (such as one called "do not delete.txt"). This will ensure the folder is not deleted when the platform is re-installed (and therefore requiring you to re-map the permissions).

Path	**Purpose**
%NQROOT%\Logs	Platform debug logs
%NQROOT%\UserFiles	Temporary location of uploaded documents prior to doc storage
%NQROOT%\Templates\IssueTrak	Location of template files for presenting operational data, sending email notifications.

IssueTrak Requirements

The application solves a simple software issue tracking problem. In this application, there are two important objects, an "issue" and an "individual." The issue holds information about the issue, the individual models information about a user in the system.

The basic requirements are as follows:

- Users of the system are authenticated using a directory managed by the application.
- Issues should be entered with a minimum of information – description, type, severity. The rest of the information should default to reasonable values.
- Issues can be related to each other. This relationship may be specified as related, duplicate, dependency. Relationships are displayed only in one direction.
- The Issue must support attachments and free text notes.
- The Issue and Individual objects should track the last updated user and last updated date/time.
- Changes to Issue status should result in an email notification.
- Changes to major objects should be audited.
- Provide the ability to import and export individuals and issues from /to external systems.
- Send arbitrary emails to individuals in the system.
- Showcase the flexibility of the platform metadata and richness of the extension model.

Next Steps

You have now set up the NetQuarry Platform and prepared your computer to create the IssueTrak application. Proceed to the next document, NetQuarry – IssueTrak Tutorial – Part2, where we will describe how to create the basics of the IssueTrak application through metadata.