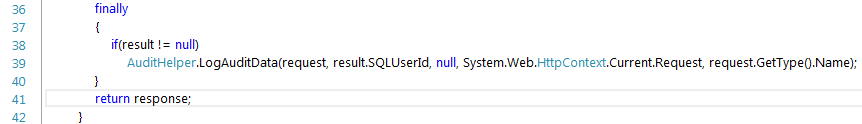
## Purpose

The purpose of the Action Audit is to track the types of actions a user takes, in the application, as well as logging the id’s of any patients who’s details a user has viewed.

The specifics of a particular action are stored in the AuditAction table of the SQL Server database that has been configured to hold all standard application data. (Currently, the database connection name for this configuration can be found in the web.config file of the Nightingale service, under the key “PhytelServicesConnName”.)

## Creating an ActionAudit Record

To add an audit record to the table, the **LogAuditData** method of the **AuditHelper** object should be called. Currently, these are called from each of the Nightingale service methods, just before the response object is returned to the caller. For example:



When this call is made, a new message queue message is created. That message is then dropped into a specific queue and will ultimately be picked up and processed by the **ASE Processor** application. That message also contains all the details for the database record, inside its message body.

The queue that holds those messages is designated by a value in the “ApplicationSettings” table of the database, under the key “AUDIT\_QUEUE”.

## ActionAudit Objects and Methods

These objects are in the Phytel.API.Common project, which is inside the API solution.

### AuditData.cs

This is the POCO data object that holds all the details about an action. This object is used to populate the body of the queue message, which in turn, gets added to the database.

#### Structure:

namespace Phytel.API.Common.Audit

{

[Serializable]

public class AuditData

{

public AuditData() { }

#region Required Properties

public string Type { get; set; }

public int AuditTypeId { get; set; }

public Guid UserId { get; set; }

public Guid ImpersonatingUserId { get; set; }

public string SourcePage { get; set; }

public string SourceIP { get; set; }

public string Browser { get; set; }

public string SessionId { get; set; }

public int ContractID { get; set; }

public DateTime EventDateTime { get; set; }

#endregion

#region Optional Properties

// patient list

[XmlArray("PatientIDList")]

[XmlArrayItem("PatientID")]

public List<string> Patients { get; set; }

public Guid EditedUserId { get; set; }

public string EnteredUserName { get; set; }

public string SearchText { get; set; }

// additional audit information

public object AdditionalAuditData { get; set; }

public string LandingPage { get; set; }

public Message Message { get; set; }

public string TOSVersion { get; set; }

public string NotificationTotal { get; set; }

public string DownloadedReport { get; set; }

#endregion

}

[Serializable]

public class Message

{

public Message()

{

}

public string Id { get; set; }

public string Text { get; set; }

public string Source { get; set; }

public string StackTrace { get; set; }

}

}

### AuditHelper.cs

* + public static void LogAuditData(IAppDomainRequest request, string sqlUserID, List<string> patientids, HttpRequest webreq, string returnTypeName)

This is the entry point to create an action audit record. It’s purpose is to create a new thread, pass its arguments through to a new method inside that new thread to handle all the processing and immediately return control back to the original thread, while the audit is being processed.

* + - IAppDomainRequest request:

the actual request object created by the service that receives the web request

* + - string sqlUserID:

the user id for the user currently making the request

* + - List<string> patientids:

A list of id’s for any patients who’s data was viewed during this request

* + - HttpRequest webreq:

The web request object generated by the original call from the browsaer to the service endpoint

* + - string returnTypeName:

The type name of the specific request object passed to the service

* + private static void AuditAsynch(IAppDomainRequest request, string sqlUserID, List<string> patientids, HttpRequest webreq, string returnTypeName)

This is the starting point for processing of the actual audit. The arguments for this method are identical to the previous method and are simply passed through, from the previous message.