

# WISDOM SECURITY TRIMMER

INTEGRATING EDM WITH LINE OF BUSINESS APPLICATIONS WITH COMPLEX SECURITY MODELS

## Document Information

<b>Title:</b>	Wisdom Security Trimmer
<b>Author:</b>	Alex Radice
<b>Current Issue &amp; Change Date:</b>	1.0 12/07/2010
<b>Filename (electronic)</b>	Wisdom Security Trimmer.docx
<b>Location (electronic):</b>	<a href="http://one.mymorse.com/sites/ac/pac/Wisdom/Research/Shared Documents/Wisdom Security Trimmer.docx">http://one.mymorse.com/sites/ac/pac/Wisdom/Research/Shared Documents/Wisdom Security Trimmer.docx</a>
<b>Owner:</b>	Wisdom Core Team
<b>Distribution:</b>	Wisdom Core Team, Wisdom Delivery Team, Wisdom Integration Partners

## Document History

Issue	Change Owner	Date	Details of Change
1.0	Alex Radice	12/07/2010	Released.

## INTRODUCTION

Many organisations are choosing to use EDRM systems as backing stores for existing line of business systems. This allows them to benefit from the auditing, search and retention of an EDRM system while allowing users to continue using the line of business systems with which they are familiar.

When Wisdom eDRM is used as a backing store in this way, access to the documents stored would normally be controlled by the Wisdom security model. Wisdom has a very flexible security model, allowing EDRM administrators to combine permissions from different Active Directory or eDirectory groups, Wisdom Teams and built in groups.

For some line of business systems which have very complicated or rapidly changing patterns of user access and restriction, it may be difficult to mirror these in Wisdom. In these circumstances it is possible to create a custom “Security Trimmer” integration component which will allow an external system to augment or override the Wisdom security model.

## REQUIREMENT BREAKDOWN

**The Security Trimmer integration point needs to allow an external system to modify the result of any Wisdom access check.**

Wisdom uses a combination of its own Access Control Lists and custom security types to decide whether a given user has access to a given item. There are then further checks made to decide what operations the user can carry out against the item. The Security Trimmer integration point will only allow **access** to be modified, **not the list of permissible operations**.

## SEPARATION OF INTEGRATION POINT, IMPLEMENTATION AND CONFIGURATION

The **Security Trimmer integration point** is a .NET interface, defined in the Wisdom.Common.Providers assembly which is installed by Wisdom.

To connect to a given external system which performs security checks a .NET class which implements this interface must be written and deployed to each Wisdom Application Server: this is the **Security Trimmer implementation**.

To notify Wisdom of the existence of the new Security Trimmer implementation a “<Trimmer>” entry must be placed in the *wisdom.config* file. This **Security Trimmer configuration** specifies:

- How the native Wisdom Security check is combined with the external check.
- Filtering rules for which areas of the file plan use the Security Trimmer.
- Additional configuration details required by the particular Security Trimmer implementation

## COMBINATION OF WISDOM AND EXTERNAL SECURITY CHECKS

Introducing an additional security check raises the question of how it will interact with the native Wisdom security check. The full range of possibilities is:

1. Grant access if either system grants access.
2. Grant access if both systems grant access.
3. Grant access if the native Wisdom security check grants access.

- Grant access if the external system grants access – if this option is used the native Wisdom security check is ignored.

Which of these options is used is defined in the configuration that references a given Security Trimmer implementation.

## SELECTIVE APPLICATION OF SECURITY TRIMMING

A typical line of business system which is integrated with Wisdom eDRM will use a particular area of the Wisdom file plan to store its documents. Security trimming should therefore only be applied to this area of the file plan. To accommodate this it is possible to specify a regular expression in the Security Trimmer configuration which is applied to the fully qualified reference of a file plan item to determine whether the Security Trimmer implementation is invoked for that item.

It is also possible to carry out more detailed filtering in the Security Trimmer implementation. For example:

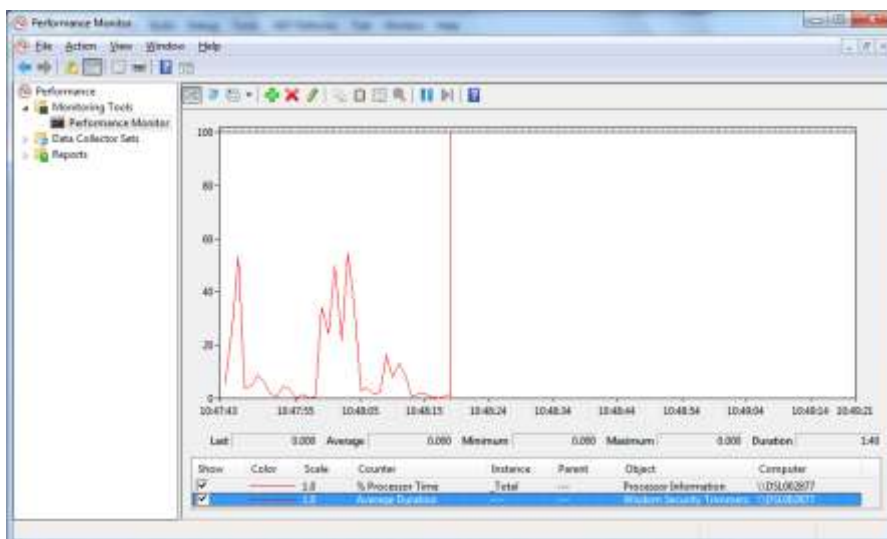
- Applying security trimming to all documents of a specified type.
- Applying security trimming only to areas of the file plan which are open.

Note however that this would have a performance impact, as the Security Trimmer implementation would need to do more work to determine whether Security Trimming should be applied to a given item

## SYSTEM PERFORMANCE CONSIDERATIONS

Checking whether a user has access to a file plan item is a very common operation. It is important that a Security Trimmer implementation performs its checks as fast as possible. To try and reduce the impact on system performance Wisdom sends requests through to the Security Trimmer implementation in batches; for more information see the [Technical Details](#) section below.

A new Windows performance counter, “Wisdom Security Trimmers – Average Duration” is provided that allows the average duration of calls to all Security Trimmer implementations to be monitored. It is recommended that individual implementations also define and log to their own performance counters. These can then be monitored and logged using standard Windows system monitoring tools such as Performance Monitor, as shown below.



It is highly recommended that Security Trimmer implementations make use of a local database cache to improve performance. The cache hit ratio should be monitored using a performance counter.

## SYSTEM AVAILABILITY CONSIDERATIONS

Security Trimmer implementations will almost always query an external system via some form of remote procedure call, such as a web service call or database lookup. Consideration should be given to how the Security Trimmer implementation should behave if the external system is unavailable. It is normally undesirable to grant access in these situations as this allows users to exploit infrastructure problems to gain access to restricted information. However it can also create a problem if unavailability of the external system totally shuts down an area of the Wisdom file plan, and causes Wisdom searches which find any item from that area of the file plan to crash.

A sensible compromise is to make the Security Trimmer implementation always grant access to a particular group of users (for example, Wisdom Global Administrators) without performing a check, so that there will still be some users who can access the file plan if the external system fails.

## TECHNICAL DETAILS

A Security Trimmer implementation must implement the following interface (defined in the assembly `Wisdom.Common.Providers`):

```
namespace Diagonal.Wisdom.Common.Providers
{
    public interface ISecurityTrimmerFactory
    {
        ISecurityTrimmer GetInstance(Dictionary<string, string> parameters);
    }
}
```

The `GetInstance` method returns an object implementing `ISecurityTrimmer` which will perform Security Trimming using the parameters specified. The parameters are those defined in the `wisdom.config` file for this implementation. The

The `ISecurityTrimmer` interface is as follows:

```
namespace Diagonal.Wisdom.Common.Providers
{
    public interface ISecurityTrimmer
    {
        IEnumerable<ISecurityTrimmerResult> Trim(IEnumerable<ISecurityTrimmerRequest> requests);
    }
}
```

When Wisdom needs the Security Trimmer implementation to perform Security Trimming it passes a batch of `ISecurityTrimmerRequest` objects to the `Trim` method, which must return one `ISecurityTrimmerResult` object per `ISecurityTrimmerRequest`.

The definitions of `ISecurityTrimmerRequest` and `ISecurityTrimmerResult` are as follows:

```
namespace Diagonal.Wisdom.Common.Providers
{
    /// <summary>
    /// Interface implemented by objects which hold the details of a request passed to a
    /// security trimmer.
    /// </summary>
    public interface ISecurityTrimmerRequest
    {
        /// <summary>
        /// Result of Wisdom security check
    }
}
```

```

/// </summary>
bool CanSee { get; }
/// <summary>
/// Gets the detailed file plan item data set. This is only available if
/// <see cref="IsDetailedDataSetAvailable"/> returns true.</summary>
/// <value>The detailed file plan item data set.</value>
DataSet DetailedDataSet { get; }
/// <summary>
/// Gets the summary details for this Document, if available. This is only populated
/// if the <see cref="IsDocumentSummaryRowAvailable"/> is set to true.
/// </summary>
/// <value>The summary details for this document.</value>
DataRow DocumentSummaryRow { get; }
/// <summary>
/// Internal Wisdom ID of file plan item
/// </summary>
Guid FilePlanItemID { get; }
/// <summary>
/// Gets a value indicating whether <see cref="DetailedDataSet"/> is populated for
/// this instance.</summary>
bool IsDetailedDataSetAvailable { get; }
/// <summary>
/// Gets a value indicating whether <see cref="DocumentSummaryRow"/> is populated for
/// this instance.</summary>
/// <value>
/// <c>true</c> if a document summary is available; otherwise, <c>false</c>.
/// </value>
bool IsDocumentSummaryRowAvailable { get; }
/// <summary>
/// Gets a value indicating whether data about the file plan item as a linked item is
/// available.</summary>
bool IsLinkedItemAvailable { get; }
/// <summary>
/// Gets a value indicating whether a folder part data row is available for this
/// instance.</summary>
/// <value>
/// <c>true</c> if folder part data is available; otherwise, <c>false</c>.
/// </value>
bool IsPartDataRowAvailable { get; }
/// <summary>
/// Gets information about the file plan item as a linked item.
/// </summary>
/// <value>The linked item data structure.</value>
LinkedItem LinkedItem { get; }
/// <summary>
/// Gets the short details for the file plan item, if available.
/// </summary>
/// <value>The short details for the file plan item.</value>
FilePlanListItem ListItem { get; }
/// <summary>
/// Gets the folder part data row.
/// </summary>
/// <value>The folder part data row.</value>
DataRow PartDataRow { get; }
/// <summary>
/// Array of the reference paths to the file plan item
/// </summary>
string[] ReferencePaths { get; }
/// <summary>
/// Gets limited information about the file plan item.
/// </summary>
/// <value>The file plan item data.</value>
/// <remarks>
/// <para>This member is always available.</para>
/// <para>The item returned will be of one of the following types:</para>
/// <list type="bullet">
/// <item><see cref="DivisionDetails"/></item>
/// <item><see cref="ClassDetails"/></item>
/// <item><see cref="EntityDetails"/></item>
/// <item><see cref="DocumentDetails"/></item>
/// </list>
/// </remarks>
FilePlanItemDetails ShortItemData { get; }
}
}

```

```
namespace Diagonal.Wisdom.Common.Providers
{
    /// <summary>
    /// Interface implmented by objects containing details of the result of a call to a class
    /// implementing the <see cref="ISecurityTrimmer"/> interface.
    /// </summary>
    public interface ISecurityTrimmerResult
    {
        /// <summary>
        /// Result of the ISecurityTrimmer security check
        /// </summary>
        bool CanSee { get; }
        /// <summary>
        /// The request to the ISecurityTrimmer class
        /// </summary>
        ISecurityTrimmerRequest Request { get; }
    }
}
```

For further technical information on the Wisdom Security Trimmer, please contact the Wisdom Support Team.