APERIO APPLICATION NOTE

September 18, 2012

DataServer Method and Record Hooks Configuration Guide

This application note describes how to configure method and record hooks in order to trigger exits from the DataServer process. These exit messages can then be consumed by external applications or extensions to the Aperio image management system.

Introduction

Aperio's DataServer software supports an extensive list of hooks. These hooks allow third-party developers the opportunity to extend and enhance the default behavior of DataServer. Most often these hooks can be used to signal other software that certain database transactions have occurred or are about to occur.

There are two types of hooks – method and record. Method hooks occur when one of DataServer's web service methods are invoked. Record hooks occur when changes are made to records within a database table.

A DataServer record hook is more granular. Instead of hooking a call to a web service method it is possible to hook every record delete. This allows a hook program to be more selective about performing an action based on a particular record deletion.

Configuring DataServer

Method and record hooks are both defined in the DataServer.exe.config file. A **section** element is defined in the **configSections** collection with the name **hooksConfiguration** as follows:

```
<configSections>

  ... Other section declarations ...

  <section name="hooksConfiguration"
           type="Aperio.DbObjects.Hooks.HooksConfigSection,Aperio.DbObjects"
  />
</configSections>
```

The **name** and **type** attributes must contain the exact values as above.

After making the above change, a **hooksConfiguration** section should be created in DataServer.exe.config file as follows:

```
<hooksConfiguration>
  <methodHooks>
    <add
        hookName="Logon_Hook"
        methodName="Logon"
        hookPath="C:\Test\bin\Debug\Test.exe"
        parametersAsFile="true"
        useShellExecute="false"
        timing="After, Before"
        enabled="false"
    />
  </methodHooks>
  <recordHooks>
    <add
        hookName ="ImageDelete_Hook"
        hookPath="C:\Test\bin\Debug\Test.exe"
        parametersAsFile="false"
        rowState="Deleted"
        tableName="Image"
    />
  </recordHooks>
</hooksConfiguration>
```

NOTE: parameter names and values are case-sensitive. **true** and **false** values are lowercased.

Any number of **recordHooks** and/or **methodHooks** can be defined inside the **hooksConfiguration** section.

Some of the parameters (**hookName**, for example) are common for both, method and record hooks. Others are specific to the hook type. Possible hook parameters are detailed below.

Parameters common between record hooks and method hooks

Parameter	Required	Valid Values	Default Value	Description
hookName	Yes	Any string	n/a	The unique name of the hook. DataServer will pass it in as one of the parameters to the hook handler.
hookPath	Yes	Any Valid UNC or mapped-drive path	n/a	The path to the executable/batch file serving as a hook handler, (useShellExecute must be set to true for anything other than .exe)
parametersAsFile	No	"true" or "false"	"false"	If set to true, then DataServer saves hook data to temporary file and passes the file name in as a command line argument. Saving data in files instead of passing directly to the handler is especially useful when working with useShellExecute = true because of the URL length limitation imposed by WIN32 API (http://support.microsoft.com/kb/263909)
enabled	No	"true" or "false"	"true"	Allows to enable/disable some of the hooks without the need to remove their definition from the config file.
useShellExecute	No	"true" or "false"	"false"	Gets or sets a value indicating whether to use the operating system shell to start the process. When you use the operating system shell to start processes, you are able to start any document (which is any registered file type associated with an executable that has a default open action). When useShellExecute is false, you are able to start only executables with the Process component.
waitForExit	No	32 bit integer ranging from 1 to 2,147,483,647	-1	The amount of time, in milliseconds, to wait for the associated process to exit. The maximum is the largest possible value of a 32-bit integer, which represents infinity to the operating system. If set to -1, then DataServer does not wait for the process to finish. The default value of -1 should be used with caution - if the hook performs some long lasting operation, it is possible for the DataServer to call it the second time before the completion of the first call. Hook's implementation must be aware of that behavior and support re-entrance.

Parameters specific to method hooks

Parameter	Required	Valid Values	Default Value	Description
methodName	Yes	See description	n/a	The name of the DataServer web service method that should trigger the hook.
timing	No	“Before” “After”	“After”	Commands DataServer when to execute the hook – before or after the method is called. It could be a combination of both (comma-separated list, e.g. “Before,After”).

Parameters specific to record hooks

Parameter	Required	Valid Values	Default Value	Description
tableName	Yes	See description	n/a	The name of the database table to monitor.
timing	Yes	“Added” “Deleted” or “Modified”	n/a	The transaction event for the table that triggers the hook. Valid values can be combined in a comma-separated list for an inclusive OR declaration, for example “Modified,Added” triggers off of both transaction types.

XML formats for the hook outputs

DataServer executes the hook when conditions match to the configuration parameters described above. When executing a hook, DataServer passes in a single command-line argument in XML format (depending on the **parametersAsFile** configuration parameter, DataServer can also save this XML to a temporary file and pass in the file name instead). Method and record hooks emit different types of XML schema, which are described next.

Method hook output XML elements

Parameter	Description
MethodHookArguments	Root element of the XML document.
HookName	The name of the hook as declared in the **hookName** attribute in the DataServer configuration file.
UserId	The user account that was logged on during execution of the method attached to the hook.
Token	The session token of the user account which executed the triggering method. The hook recipient can use it to communicate back to DataServer under that user's identity.
DataBefore	The XML input parameters provided during execution of the DataServer method.
DataAfter	The XML output of the DataServer method call. The detailed structure will be specific to the method being watched.

Sample XML output from a method hook triggered by a PutRecordData method call updating a Slide record's *BarcodeId* field:

```
<MethodHookArguments>
    <Token>f5gHzmgSn6wTL8T6s4XDCfojh... wVOgwRUdRQsyKmmC2Sw==</Token>
    <UserId>3</UserId>
    <HookName>PutSlideData_Hook</HookName>
    <DataBefore>
        <PutRecordData>
            <TableName>Slide</TableName><Id>1021</Id>
            <DataRow>
                <BarcodeId>ABC-123</BarcodeId>
            </DataRow>
        </PutRecordData>
    </DataBefore>
    <DataAfter>
        <PutRecordDataResponse>
            <PutRecordDataResult>
                <ASResult>0</ASResult><ASMessage></ASMessage>
            </PutRecordDataResult>
            <Id>1021</Id>
        </PutRecordDataResponse>
    </DataAfter>
</MethodHookArguments>
```

Record hook output XML elements

Parameter	Description
RecordHookArguments	Root element of the XML document.
HookName	The name of the hook as declared in the **hookName** attribute in the DataServer configuration file.
UserId	The user account that was logged on during execution of the transaction on the table.
Token	The session token of the user account which executed the triggering method. The hook recipient can use it to communicate back to DataServer under that user's identity.
ModifiedColumns	A list of column names for those fields that were modified during the table transaction. Contains a collection of Column elements naming each.
RowState	The type of transaction that caused the hook to trigger. Possible values are "Added", "Modified" or "Deleted", exclusively.
TableName	The name of the table declared in the configuration's **tableName** attribute of the hook.
Data/DataRow	A complete record row containing the Name/Value elements representing every table field (including user defined) after the transaction completed.

Sample XML output from a Slide table transaction updating a Slide record's *BarcodeId* field:

```
<RecordHookArguments>
    <Token>f5gHzmgSn6wTL8T6s4XDCfojh... wVOgwRUdRQsyKmmC2Sw==</Token>
    <UserId>3</UserId>
    <HookName>SlideTable_Hook</HookName>
    <ModifiedColumns>
        <Column>BarcodeId</Column>
    </ModifiedColumns>
    <RowState>Modified</RowState>
    <TableName>Slide</TableName>
    <Data>
        <DataRow>
            <Id>1021</Id>
            <BarcodeId>ABC-123</BarcodeId>
            <DataGroupId>7</DataGroupId>

            ...

            <ColumnStainer>VS_1025</ColumnStainer>
        </DataRow>
    </Data>
</RecordHookArguments>
```

Part Number/Revision: MAN-0227, Revision A
Date: 18 September 2012

This document applies to software versions Release 10.2 and later.

User Resources

For the latest information on Aperio products and services, please visit the Aperio website at: http://www.aperio.com.

Disclaimers

Use normal care in maintaining and using the eSlide Manager™ servers. Interrupting network connections or turning off the eSlide Manager and DSR servers while they are processing data (such as when they are analyzing digital slides or generating an audit report) can result in data loss.

This Application Note is not a substitute for the detailed operator training provided by Aperio or for other advanced instruction. Aperio Field Representatives should be contacted immediately for assistance in the event of any instrument malfunction. Installation of hardware should only be performed by a certified Aperio Service Engineer.

ImageServer is intended for use with the SVS file format (the native format for eSlides created by scanning glass slides with the ScanScope scanner). Educators will use Aperio software to view and modify eSlides in Composite WebSlide (CWS) format.

Aperio products are FDA cleared for specific clinical applications, and are intended for research and educational use for other applications. They are not approved by the FDA for primary diagnosis. For clearance updates, visit www.aperio.com

Trademarks and Patents

ScanScope is a registered trademark and ImageServer, TMALab, ImageScope, eSlideShare and eSlide Manager are trademarks of Aperio Technologies, Inc. All other trade names and trademarks are the property of their respective holders.

Aperio products are protected by U.S. Patents: 6,711,283; 6,917,696; 7,035,478; 7,116,440; 7,257,268; 7,428,324; 7,457,446; 7,463,761; 7,502,519; 7,518,652; 7.602.524, 7,646,496; 7,738,688 and licensed under one or more of the following U.S. Patents: 6,101,265; 6,272,235; 6,522,774; 6,775,402; 6,396,941; 6,674,881; 6,226,392; 6,404,906; 6,674,884; and 6,466,690.

Contact Information

Headquarters	Europe Office	Asia Office
Aperio 1360 Park Center Drive Vista, CA 92081 United States	Aperio UK Ltd Prama House 167 Banbury Road Summertown Oxford OX2 7HT United Kingdom	Aperio KK UZ Building 5F 3-3-17, Surugadai Kanda, Chiyoda-ku Tokyo, Japan 101-0062
Tel: 866-478-4111 (toll free) Fax: 760-539-1116	Tel: +44 (0) 1865 339651 Fax: +44(0) 1865 339301	Tel: +81-3-3259-5255 Fax: +81-3-3259-5256
Customer Service: Tel: 866-478-4111 (toll free) Email: info@aperio.com	**Customer Service:** Tel: +44 (0) 1865 339651 Email: europeinfo@aperio.com	**Customer Service:** Email: asiainfo@aperio.com
Technical Support: US/Canada Tel: 1 (866) 478-3999 (toll free) Direct International Tel: 1 (760) 539-1150 US/Canada/Worldwide Email: support@aperio.com	**Technical Support:** Direct International Tel: 1 (760) 539-1150 Email: europesupport@aperio.com	**Technical Support:** Direct International Tel: 1 (760) 539-1150 Email: asiasupport@aperio.com