Overview of Shredded Storage in SharePoint 2013

Bill Baer

Microsoft Corporation

Reviewer: Rob Barker, NetApp

July 2013

Applies to: SharePoint Foundation 2013, SharePoint Server 2013

Summary: This white paper provides an overview of Shredded Storage in SharePoint Server 2013 and the evolution of the SharePoint Products Storage model.

Contents

Overview

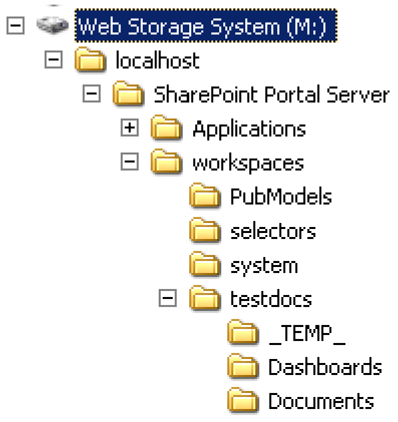
Shredded Storage is a new storage model implementation in SharePoint Server 2013 used to provide smoother I/O patterns, improve data transfer performance, and reduce storage utilization when using historical versions with SharePoint.

This white paper provides a background of SharePoint products storage evolution and the implementation specifics and benefits of Shredded Storage in SharePoint 2013.

SharePoint Portal Server 2001

Microsoft
SharePoint Portal Server 2001
Please choose an option to the right.
Server Installation
Help
Client Installation
Readme
Tours
Exit

SharePoint Portal Server 2001 represented the first commercially available version of SharePoint and utilized a unique new storage model based on the Web Storage System originally implemented in Exchange Server 2000. The Web Storage System (ironically, "WSS") implemented a hierarchical folder model for storing unstructured content (i.e. Word Documents, PowerPoint Presentations, etc.) [see Figure 1 Web Storage System] with support for accessing and updating the content through a set of APIs and Internet protocols.



```
⊟ Web Storage System (M:)
    ⊟ localhost
        ⊟ SharePoint Portal Server
            ⊞ Applications
            ⊟ workspaces
                  PubModels
                  selectors
                  system
                ⊟ testdocs
                      _TEMP_
                      Dashboards
                      Documents
```

Figure 1 Web Storage System

The Web Storage System also implemented a store-level event model that supported both synchronous and asynchronous processing in addition to a lightweight workflow engine [see Figure 2 Web Storage System Store-Level Event Model].

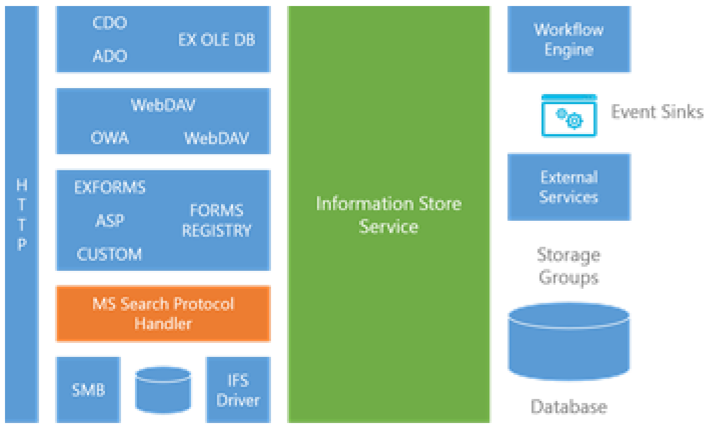


Figure 2 Web Storage System Store-Level Event Model

Definitions

CDO (Collaborative Data Objects)

CDO provides access to Outlook-compatible objects through a COM-based API. For example, an application can connect to a MAPI store and then perform operations against that store, including creating and processing calendar items, and resolving and handling mail recipients.

IFS (Installable File System)

The installable file system (IFS) provides access to the Microsoft Web Storage System that SharePoint Portal Server uses.

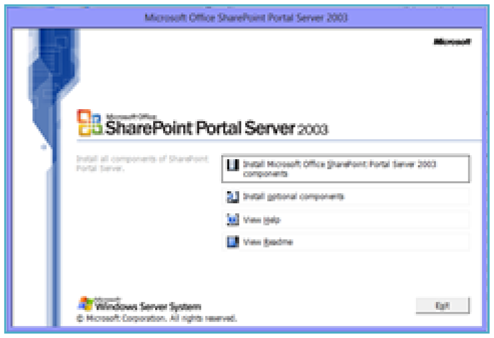
In SharePoint Portal Server 2001, IFS access is used for:

- Read-only access to the document library
- Microsoft FrontPage Server Extensions
- Web Storage System development through IFS

SMB (Server Message Block)

SMB is an application-layer network protocol commonly used for providing shared access to files, printers, and serial ports.

SharePoint Portal Server 2003


Microsoft Office SharePoint Portal Server 2003
Microsoft
Microsoft Office
SharePoint Portal Server 2003
Install all components of SharePoint Portal Server.
Install Microsoft Office SharePoint Portal Server 2003 components
Install optional components
View Help
View Readme
Microsoft Windows Server System
© Microsoft Corporation. All rights reserved.
Exit

SharePoint Portal Server 2003 fundamentally changed the semantics of BLOB (binary large object) storage by routing the binary stream associated with a file to one or more SQL Server content databases, which in addition to the file stored an individual site's structured data. Unlike SharePoint Portal Server 2001, SharePoint Portal Server 2003 stored all end-user data in SQL Server databases, providing several advantages over the Web Storage System, such as:

- Storing list data, documents, and associated metadata in normalized tables
- Support for transactional updates of documents and document metadata
- A unified backup solution for documents and document metadata

The Web Storage System supported one database per site and table per list, the new relational database model in SharePoint Portal Server 2003 implemented a fixed database schema and number of databases per server to enable more effective horizontal scaling capabilities.

The primary storage tables in SharePoint Portal Server 2003 included the Sites, Docs, Lists, Links, and WebParts tables.

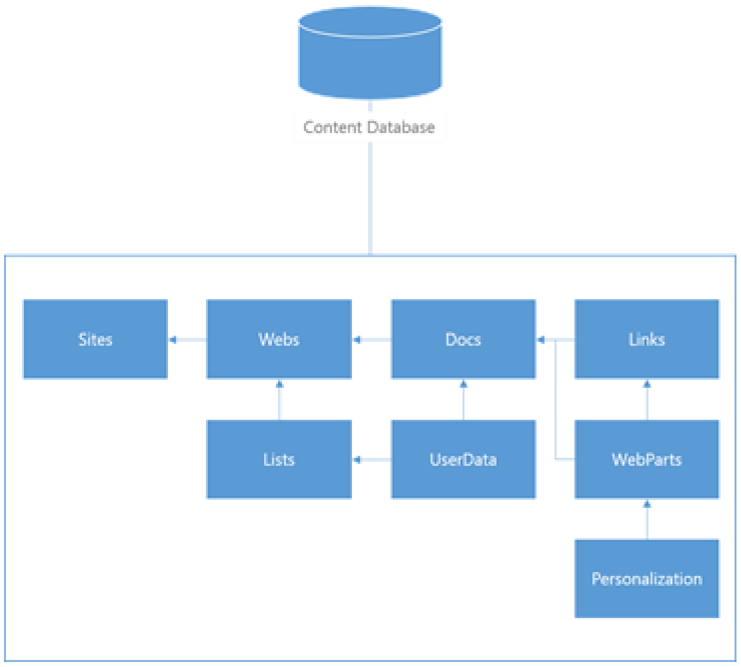


Figure 3 SharePoint Content Database Primary Storage Tables

DBO.SITES

In SharePoint Portal Server 2003, the Sites table stores settings that apply to individual site collections representing the top-level site of each site collection, including the root site and My Site as related to the portal site. Subordinate objects such as Webs and their corresponding settings are stored in the Webs tables.

DBO.DOCS

The Docs table stores all documents within their respective site collections such as documents in document libraries, attachments, list nodes, and customized users pages.

The Content column in Docs is defined to store unstructured content generated by users and is based on the image data type. The image data type, removed from future versions of SQL Server, was a variable-length binary data from 0 through 2^31-1 (2,147,483,647) bytes (i.e. 2GB).

DBO.LISTS

The Lists table contains a row for each list of all the sites in the database. This table contains settings for each list, specifying which lists or document libraries are included in the sites.

DBO.LINKS

The Links table contains links used in link fix-up to recalculate links.

DBO.WEB PARTS

The Web Parts table contains information about all the Web Parts and list views used in the sites. Web Part personalization information are maintained in the Personalization table.

SharePoint Portal Server 2003 uses foreign key relationships into tables and added two additional databases, the Profile and Services databases. The Profile database stores personal profiles and audience definitions for targeting of Web Parts and content, and the Services database supports search and indexing as well as subscriptions and subscription results.

Type your 25-character Product Key. The key is located on a card in your product carton or on your license agreement.

Office SharePoint Server 2007

Office SharePoint Server 2007 carries forward the relational database storage model of SharePoint Portal Server 2003 with notable exceptions, including changes to the content database schema as related to the storage of unstructured content.

External BLOB Storage

Office SharePoint Server 2007 introduced new methods to support the externalization of user content (BLOBs) or unstructured data through External BLOB Storage. External BLOB Storage in Office SharePoint Server 2007 runs in parallel to the SharePoint content databases, enabling unstructured content to reside on alternate data stores with the structured content, such as site data, residing within the content database(s). To coordinate the separate data stores, a COM interface is necessary and is implemented on servers where Office SharePoint Server 2007 is installed and uses basic semantics to recognize save and open commands that invoke redirection to BLOB storage in the event a BLOB data stream requires updating. The implemented COM interface in External BLOB Storage is referred to as a provider (ISPExternalBinaryProvider) which is installed and registered on each Web server.

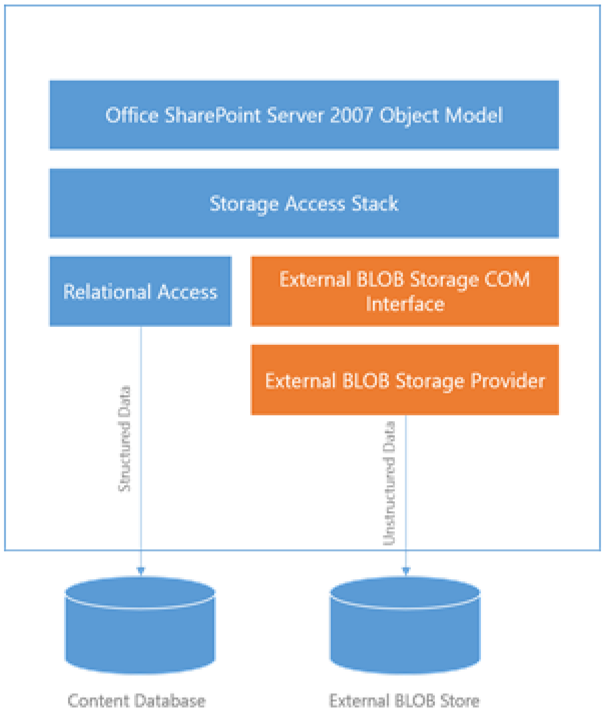

Office SharePoint Server 2007 Object Model
Storage Access Stack
Relational Access
External BLOB Storage COM Interface
External BLOB Storage Provider
Structured Data
Unstructured Data
Content Database
External BLOB Store

Figure 4 External BLOB Storage

Microsoft SharePoint Server 2010
Prepare
Review hardware and software requirements
Read the installation guide
Read the upgrade guide
Install
Install software prerequisites
Install SharePoint Server
Other Information
Visit Windows Update
Visit product website
Exit

SharePoint Server 2010

SharePoint Server 2010 maintains the relational database storage model of Office SharePoint Server 2007 and further modifies content database schema in addition to adding support for new BLOB externalization solutions.

External BLOB Storage

SharePoint Server 2010 continues to provide support for External BLOB Storage; however, it was deprecated in favor of a new unstructured data storage solution, Remote BLOB Storage.

Remote BLOB Storage

In response to deprecating support for External BLOB Storage, SharePoint Server 2010 introduces support for Remote BLOB Storage that leverages built-in SQL Server 2008 capabilities. Remote BLOB Storage is a SQL Server library API set that is provided as an add-on feature pack for SQL Server 2008 R2, SQL Server 2008 or SQL Server 2008 R2 Express. Remote BLOB Storage provides two separate solutions, a FILESTREAM provider that enables basic storage of unstructured content on either the file system of a local or remote database server and an interface to allow third-party developers to develop providers for the externalization of unstructured data through Remote BLOB Storage.

Remote BLOB Storage provides a similar solution to handling unstructured data as External BLOB Storage; however, it supports new levels of overall granularity. Whereas External BLOB Storage is a COM-based farm level implementation, Remote BLOB Storage is a .NET-based database level implementation, meaning it can be implemented for a certain subset of content, but not other content. With Remote BLOB Storage, SQL Server and SharePoint Server 2010 jointly manage the data integrity between the database records and contents of the RBS external store on a per-database basis.

A native RBS provider is made available through FILESTREAM. FILESTREAM, like many providers, implements the BLOB Store abstract class in the Client Library in order to provide BLOB operation functionality to the client application (SharePoint 2010). The FILESTREAM provider utilizes the SQL Server FILESTREAM technology to store BLOBs as files in the NTFS file system. For more details on the FILESTREAM technology, see the FILESTREAM Storage in SQL Server 2008 whitepaper.

There are two possible implementations of the FILESTREAM provider. They are the ***Local*** FILESTREAM provider and the ***Remote*** FILESTREAM provider. The FILESTREAM provider implements a database FILESTREAM file group in order to essentially turn the SQL Server NTFS file system into a BLOB store.

When you deploy the ***Local*** FILESTREAM provider, the FILESTREAM file group is created directly in the database that is being RBS-enabled. This means that the same instance of SQL Server that is processing requests from the client application database is also acting as a BLOB store (see Figure 5 Local FILESTREAM Provider).

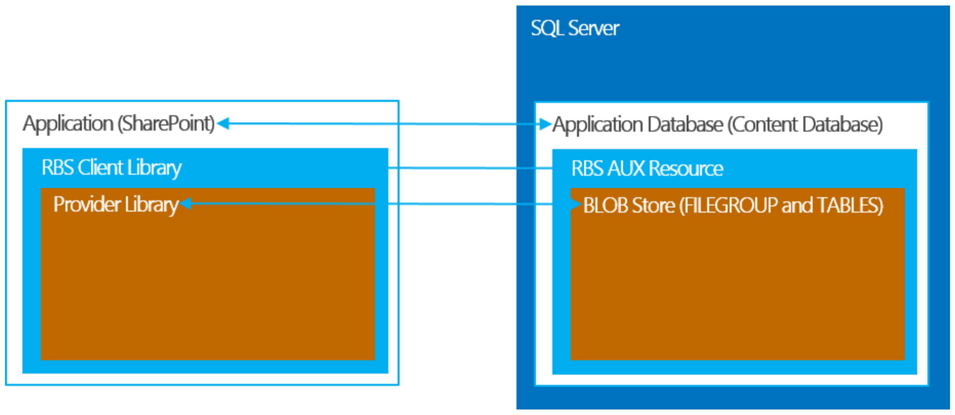


Figure 5 Local FILESTREAM Provider

Remote FILESTREAM supports creation of the FILESTREAM File Group in a content database separate from that where RBS is enabled or in a content database residing on a separate instance of Microsoft SQL Server. Using a separate server enables such scenarios as providing a dedicated server to service RBS BLOB Store requests, whereas the server hosting the associated content database can be dedicated to application processing. This implementation scenario facilitates those environments where improvements in application scalability are necessary (see Figure 6 Remote FILESTREAM Provider).

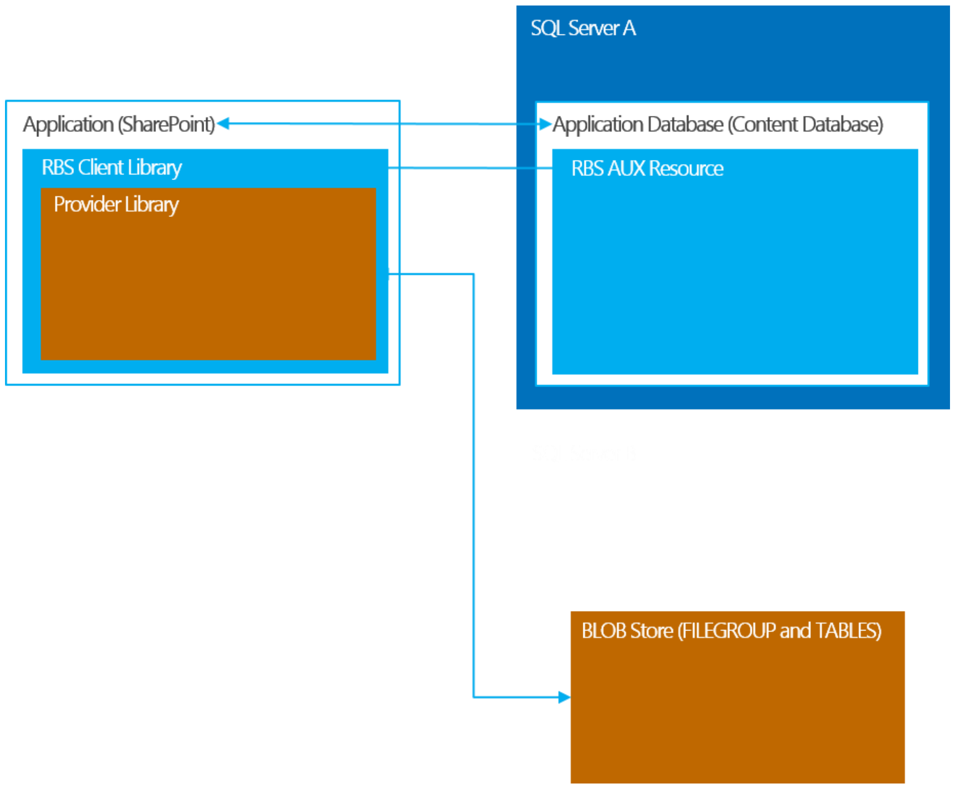


Figure 6 Remote FILESTREAM Provider

File Synchronization via SOAP over HTTP Protocol (MS-FSSHTTP)

SharePoint Server 2010 introduces new protocols to improve overall File/Save efficiency through File Synchronization via SOAP over HTTP Protocol (MS-FSSHTTP), also known as Cobalt. Implementation of MS-FSSHTTP improved over-the-network performance when users opened and saved documents back to SharePoint Server 2010 from Office 2010 clients.

Through MS-FSSHTP support users send only the compressed differentiation, or delta, of the file back to SharePoint when editing and saving. For example, if a user opens a 5 MB Word file and applies updates totaling 150 KB, only those 150 KB – or less, since it will compress it – will be sent back to the server.

The File Synchronization via SOAP over HTTP Protocol enables one or more protocol clients, such as Office 2010, to synchronize changes done on shared files stored in SharePoint Server 2010 (the protocol server). This protocol enables a protocol client to call a request that allows for the upload or download of file changes, along with related metadata changes, to or from a single protocol server. In addition, MS-FSSHTTP processes different types of locking operation requests sent by a client that allow for uploads to be done while preventing merge conflicts on the shared resource. Each file has one or more partitions associated with it and these partitions can be empty or contain binary file contents, information related to file coauthoring, or contents that are specific to a file format. The data in each partition can be synchronized independently by using MS-FSSHTTP.

File Download Semantics

For a download file request, the protocol client sends a download request to the protocol server for all the contents of a specific partition of a file specified by a URL. If the file exists on the protocol server, the protocol server responds with the requested content or partition data.

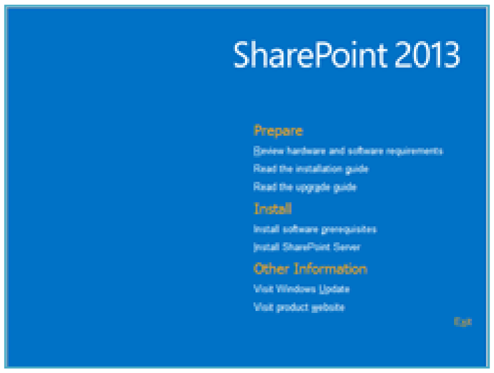
File Upload Semantics

For an upload file request, the protocol client sends an upload request to the protocol server indicating the data that has changed that needs to be uploaded. The protocol client can also send an upload request for changes done in the partitions associated with a file at a given URL. The server responds with success or failure for that update.

Coauthoring Semantics

MS-FSSHTTP support also enables multi-user authoring. For the Office client, it enables co-authoring for Word and PowerPoint. For the SharePoint Office Web Applications, it enables Excel and OneNote co-authoring.

In using the Office 2010 client with the Office Web Apps, because multiple clients can coauthor a file, if two or more clients sent an upload request simultaneously, all requests except the first one fail with a coherency error. If the upload request fails with a coherency error, the protocol client sends a download request to get the latest changes to the file from the protocol server. The protocol client automatically merges the latest changes with its local version of the file. If the protocol client is unable to do an automatic merge, it exposes the merge conflict to the user and lets the user do a manual merge. The protocol client then sends another upload request to upload the merged version of the file to the server. The upload request succeeds if the file has not been updated by another client since the last download request made by the current client.


SharePoint 2013
Prepare
Review hardware and software requirements
Read the installation guide
Read the upgrade guide
Install
Install software prerequisites
Install SharePoint Server
Other Information
Visit Windows Update
Visit product website
Exit

SharePoint Server 2013

SharePoint Server 2013 maintains the relational database storage model for unstructured content and improves overall IO. Support for External BLOB Storage is removed in SharePoint Server 2013 while support for Remote BLOB Storage is continued.

Shredded Storage

Shredded Storage is a new data platform improvement provided with SharePoint 2013 related to the management of large binary objects (i.e. BLOBS such as Microsoft PowerPoint Presentations, Microsoft Word Documents, etc.). Shredded Storage capabilities are also implemented on non-Microsoft Office file formats.

Shredded Storage both improves I/O and reduces compute utilization when making incremental changes to documents or storing documents in SharePoint 2013. Shredded Storage builds upon the Cobalt (i.e. File Synchronization via SOAP over HTTP (MS-FSSHTTP)) protocol introduced in SharePoint 2010.

In SharePoint 2010, when you save a document opened with the Office 2010 client, only the incremental change to the document is submitted over the network from the client to the server (see File Synchronization via SOAP over HTTP Protocol (MS-FSSHTTP) above). However, the document is then coalesced on the Web server, requiring a full read from the database server, and subsequently the new file, inclusive of the change, is written to the database server.

Shredded Storage is designed to ensure the write cost of updating a document is proportional to the size of the change, and not of the file itself. In core collaborative use case scenarios where historical versioning is enabled, Shredded Storage can reduce overall data storage requirements an estimated 30-40% through ensuring that only the BLOB partition (chunk) associated with the change is updated. In order to support this scenario, SharePoint 2013 stores content as a collection of independent BLOBs (Shredded Storage). When shredded, the data associated with a file such as Document.docx is distributed across a set of BLOBs associated with the file. The independent BLOBS are each assigned a unique ID (offset) to enable reconstruction in the correct order when requested by a user.

In SharePoint 2010, when a file is uploaded to a Document Library/List, a single row is created in AllDocStreams to host the BLOB associated with the upload. On subsequent edits to the file, only the changed bytes (incremental change) are sent to the Web server across the network, reducing the clients overall bandwidth utilization. However, in order to coalesce the changes, the file is read from the database server by the web server where the merge occurs and the file sent back to the database server for storage. In SharePoint 2010, this process improved the reliability of file I/O operation; however, the Web server incurred a penalty as the result of the change due to the need to coalesce the file. Shredded

Storage improves on the SharePoint 2010 model by breaking an individual BLOB into "shredded BLOBs" that are stored in a new database Table, DocStreams. Each BLOB contains a numerical Id representative of the source BLOB when coalesced. When a client updates a file, only the shredded BLOB that corresponds to the change is updated with the update occurring on the database server as opposed to the Web server. As a result, File IO operations are reduced by ~2× when compared to MS-FSSHTTP in SharePoint 2010 and the storage footprint reduced.

For example, suppose a user is working with a 10 MB Microsoft PowerPoint Presentation and makes a change — either adding a new slide, removing a slide, modifying attributes, etc.—and saves the file back to the document library where it was initially accessed. In this scenario only the portion of the file related to the change is written to the database.

Example:

User A opens a 10-MB PowerPoint Presentation. She modifies its content by adding a new slide and/or updating the presentation's attributes and properties, and she subsequently saves the file back to the originating Document Library. In this example only the portion of the file related to the change is written to the data store.

Schema Changes

Shredded Storage implements a number of schema changes in order to support the new file storage model. Those schema changes are documented below; however, they are intended only to describe the changes as implemented by Shredded Storage. The SharePoint 2013 schema should not be modified or otherwise manipulated, as documented in the Knowledge Base article http://support.microsoft.com/kb/841057.

dbo.DocStreams

In SharePoint 2010, dbo.AllDocStreams stored the document stream and related data for documents with content streams. In SharePoint Server 2013, dbo.DocStreams replaces dbo.AllDocStreams, where each row stores a portion of the BLOB.

The improved protocols associated with Shredded Storage identify the rows (in the new DocStreams table) necessary to be updated to support the change and updates the BLOB associated with that change in the corresponding row.

Several new columns are present in the DocStreams table that represent a shredded BLOB including:

- BSN: The BLOB Sequence Number (BSN) of the stream binary piece.
- Data: Contains a subset of the binary data of the stream binary piece unless the stream binary piece is stored in Remote BLOB Storage.

- Offset: The offset into the stream binary piece where this subset data belongs.

- Length: The size, in bytes, of this subset data of the stream binary piece.

- RbsId: If this stream binary piece is stored in remote BLOB storage, this value MUST contain the remote BLOB storage identifier of a subset of the binary data of the stream binary piece. Otherwise it MUST be NULL.

dbo.DocsToStreams

- A new DocToStreams table contains a pointer to a corresponding row in dbo.DocStreams. The BLOB Sequence Number (BSN) is used to manage the BLOB sequence across dbo.AllDocVersions, dbo.DocsToStreams, and dbo.DocStreams. NextBSN is used to manage the last BSN for each BLOB.

dbo.AllDocs

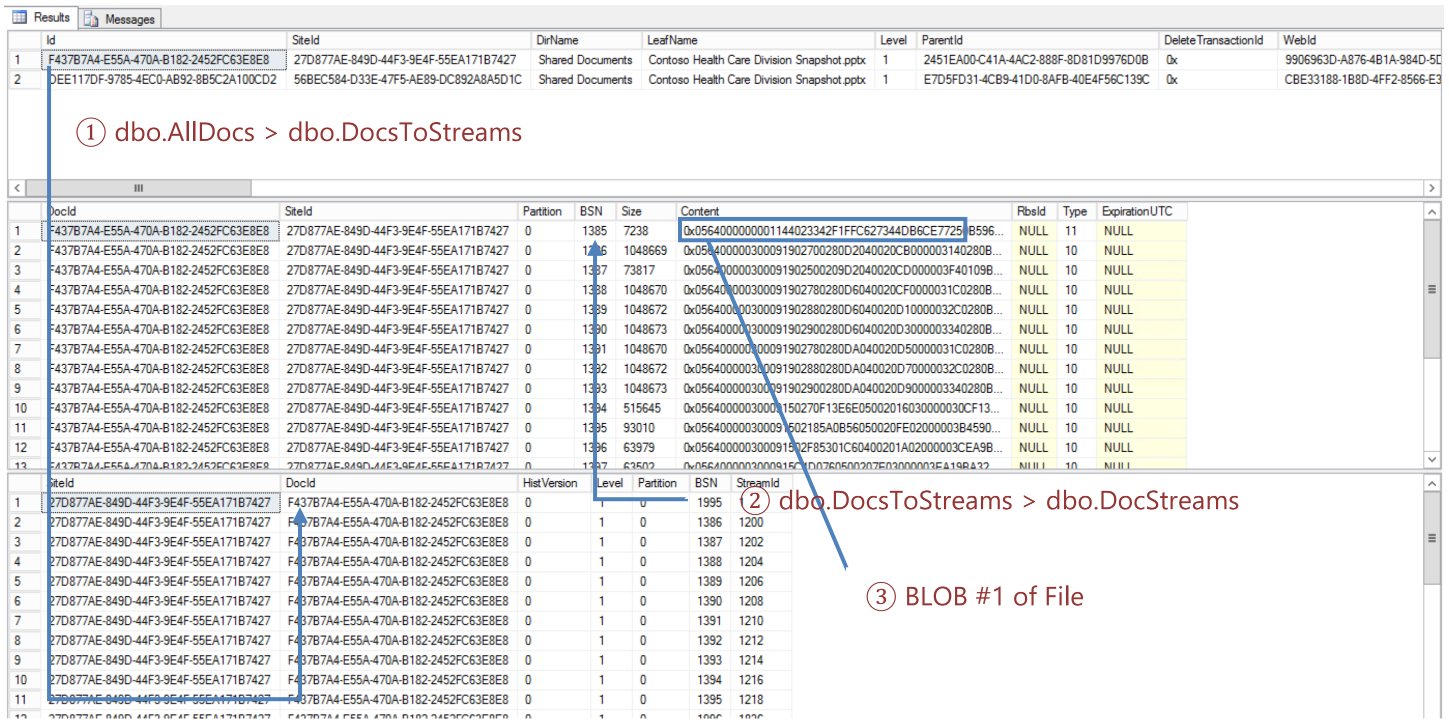
- dbo.AllDocs contains a single row per file, similar to SharePoint Server 2010.

dbo.AllDocVersions

- dbo.AllDocVersions contains one or more rows per file and one row per file version.

BLOB Storage Semantics

For a normalized single-document initial upload, the BLOB access pattern is dbo.AllDocs/dbo.AllDocVersions > dbo.DocsToStreams > dbo.DocStreams as represented in Figure 7 Schema (Initial File Upload / Un-versioned File) below.


Results
Messages
① dbo.AllDocs > dbo.DocsToStreams
② dbo.DocsToStreams > dbo.DocStreams
③ BLOB #1 of File

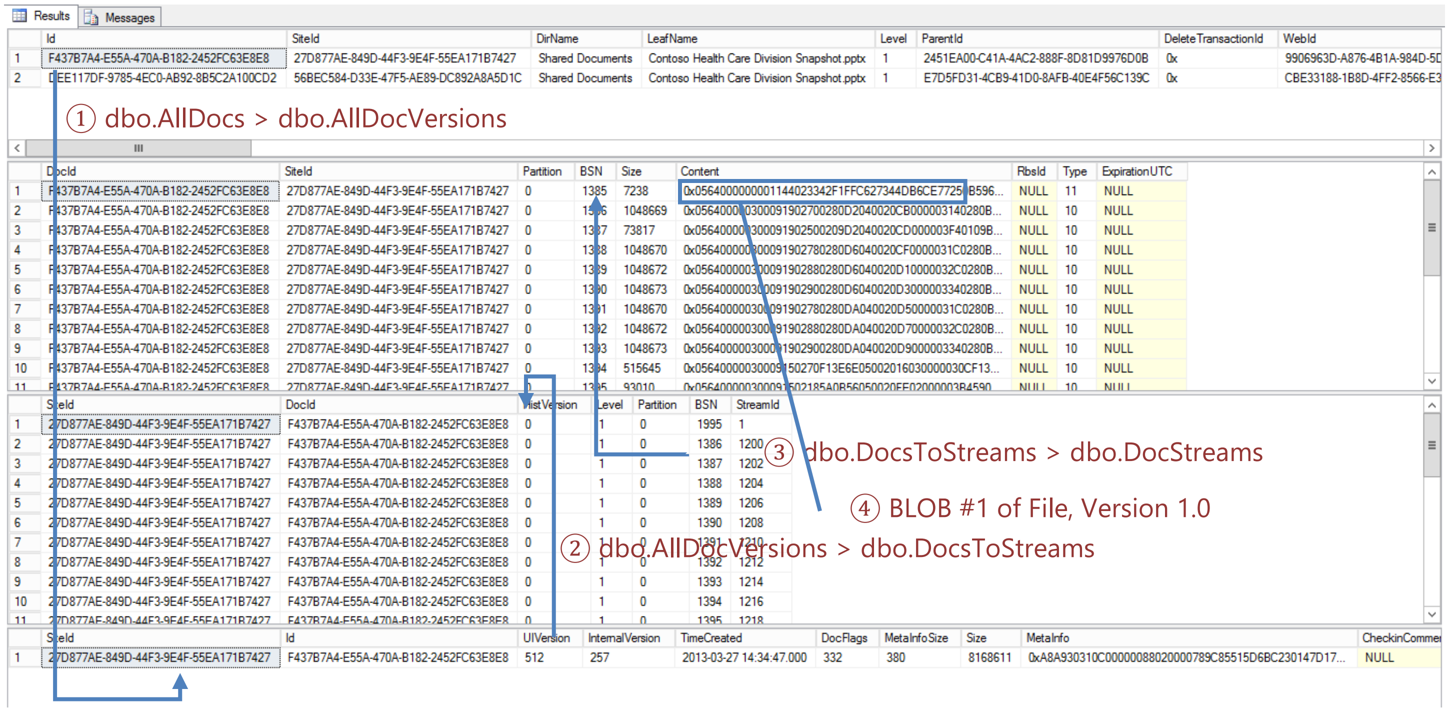
	Id	SiteId	DirName	LeafName	Level	ParentId	DeleteTransactionId	WebId
1	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	Shared Documents	Contoso Health Care Division Snapshot.pptx	1	2451EA00-C41A-4AC2-888F-8D81D9976D0B	0x	9906963D-A876-4B1A-984D-5D
2	DEE117DF-9785-4EC0-AB92-8B5C2A100CD2	56BEC584-D33E-47F5-AE89-DC892A8A5D1C	Shared Documents	Contoso Health Care Division Snapshot.pptx	1	E7D5FD31-4CB9-41D0-8AFB-40E4F56C139C	0x	CBE33188-1B8D-4FF2-8566-E3

	DocId	SiteId	Partition	BSN	Size	Content	RbsId	Type	ExpirationUTC
1	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1385	7238	0x056400000001144023342F1FFC627344DB6CE7725[illegible]B596...	NULL	11	NULL
2	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1386	1048669	0x05640000030009190270028 0D2040020CB000003140280B...	NULL	10	NULL
3	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1387	73817	0x0564000003000919025002 09D2040020CD000003F40109B...	NULL	10	NULL
4	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1388	1048670	0x05640000030009190278028 0D6040020CF0000031C0280B...	NULL	10	NULL
5	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1389	1048672	0x05640000030009190288028 0D6040020D10000032C0280B...	NULL	10	NULL
6	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1390	1048673	0x05640000030009190290028 0D6040020D3000003340280B...	NULL	10	NULL
7	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1391	1048670	0x05640000030009190278028 0DA040020D50000031C0280B...	NULL	10	NULL
8	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1392	1048672	0x05640000030009190288028 0DA040020D70000032C0280B...	NULL	10	NULL
9	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1393	1048673	0x05640000030009190290028 0DA040020D9000003340280B...	NULL	10	NULL
10	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1394	515645	0x05640000030009150270F13E6E05002016030000030CF13...	NULL	10	NULL
11	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1395	93010	0x056400000300091502185A0B56050020FE02000003B4590...	NULL	10	NULL
12	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1396	63979	0x056400000300091502F85301C60400201A02000003CEA9B...	NULL	10	NULL

	SiteId	DocId	HistVersion	Level	Partition	BSN	StreamId
1	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1995	[illegible]
2	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1386	1200
3	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1387	1202
4	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1388	1204
5	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1389	1206
6	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1390	1208
7	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1391	1210
8	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1392	1212
9	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1393	1214
10	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1394	1216
11	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1395	1218

Figure 7 Schema (Initial File Upload / Un-versioned File)

For a normalized single document with versioning enabled, the BLOB access pattern is dbo.AllDocs/dbo.AllDocVersions > dbo.AllDocVersions/dbo.DocsToStreams > dbo.DocsToStreams > dbo.DocStreams as represented in the Figure 8 Schema (Versioned File) below.


Results
Messages
① dbo.AllDocs > dbo.AllDocVersions
② dbo.AllDocVersions > dbo.DocsToStreams
③ dbo.DocsToStreams > dbo.DocStreams
④ BLOB #1 of File, Version 1.0

	Id	SiteId	DirName	LeafName	Level	ParentId	DeleteTransactionId	WebId
1	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	Shared Documents	Contoso Health Care Division Snapshot.pptx	1	2451EA00-C41A-4AC2-888F-8D81D9976D0B	0x	9906963D-A876-4B1A-984D-5D
2	DEE117DF-9785-4EC0-AB92-8B5C2A100CD2	56BEC584-D33E-47F5-AE89-DC892A8A5D1C	Shared Documents	Contoso Health Care Division Snapshot.pptx	1	E7D5FD31-4CB9-41D0-8AFB-40E4F56C139C	0x	CBE33188-1B8D-4FF2-8566-E3

	DocId	SiteId	Partition	BSN	Size	Content	RbsId	Type	ExpirationUTC
1	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1385	7238	0x056400000000001144023342F1FFC627344DB6CE77250B596...	NULL	11	NULL
2	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1386	1048669	0x0564000003000919027002 80D2040020CB000003140280B...	NULL	10	NULL
3	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1387	73817	0x05640000030009190250020 9D2040020CD000003F40109B...	NULL	10	NULL
4	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1388	1048670	0x056400000300091902780280D6040020CF0000031C0280B...	NULL	10	NULL
5	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1389	1048672	0x056400000300091902880280D6040020D10000032C0280B...	NULL	10	NULL
6	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1390	1048673	0x056400000300091902900280D6040020D30000033 40280B...	NULL	10	NULL
7	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1391	1048670	0x056400000300091902780280DA040020D50000031C0280B...	NULL	10	NULL
8	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1392	1048672	0x056400000300091902880280DA040020D70000032C0280B...	NULL	10	NULL
9	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1393	1048673	0x056400000300091902900280DA040020D90000033 40280B...	NULL	10	NULL
10	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1394	515645	0x05640000030009150270F13E6E0500201603000003 0CF13...	NULL	10	NULL
11	F437B7A4-E55A-470A-B182-2452FC63E8E8	27D877AE-849D-44F3-9E4F-55EA171B7427	0	1395	93010	0x05640000030009150218 5A0B56050020FF02000003B4590	NULL	10	NULL

	SiteId	DocId	HistVersion	Level	Partition	BSN	StreamId
1	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1995	1
2	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1386	1200
3	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1387	1202
4	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1388	1204
5	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1389	1206
6	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1390	1208
7	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1391	1210
8	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1392	1212
9	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1393	1214
10	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1394	1216
11	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	0	1	0	1395	1218

	SiteId	Id	UIVersion	InternalVersion	TimeCreated	DocFlags	MetaInfoSize	Size	MetaInfo	CheckinCommen
1	27D877AE-849D-44F3-9E4F-55EA171B7427	F437B7A4-E55A-470A-B182-2452FC63E8E8	512	257	2013-03-27 14:34:47.000	332	380	8168611	0xA8A930310C0000008802000078 9C85515D6BC230147D17...	NULL

Figure 8 Schema (Versioned File)

Reading and Writing BLOB Data

FileReadChunkSize

SharePoint 2010 introduced a new FileReadChunkSize property as a control associated with the BLOB cache which enabled a server farm administrator to control the size of incremental reads when a client requested a file.

The BLOB Cache was particularly useful when serving rich media from SharePoint. In such a scenario, the FileReadChunkSize property could be adjusted to serve files smaller than the default FileReadChunkSize (100 KB) in a single SQL Server round trip and files up to the *LargeFileChunkSize* (5 MB) served directly from SQL Server without disk buffering, resulting in low latency.

Another advantage that the BLOB cache provides is HTTP range request support. This enables a browser (or other client application) to request pieces of a file instead of the entire file. For example, if a browser only needs the last 1 MB of a 10 MB file, it can make a range request and the cache will serve only the last 1 MB. Without the BLOB cache, SharePoint Server ignores the HTTP range request and serves all 10 megabytes. The BLOB cache helps increase throughput by reducing unnecessary network load.

The FileReadChunkSize property, while accessible in SharePoint 2013, is not implemented and is instead provided through the new FileWriteChunkSize property. The FileReadChunkSize property is provided for backward compatibility.

FileWriteChunkSize

SharePoint 2013 provides access to a new FileWriteChunkSize property that replaces the FileReadChunkSize property in SharePoint 2010 that enables control of the target size of a shredded BLOB. The size of the partitioned BLOB can be configured by using Windows PowerShell or Server Object Model APIs.

The FileWriteChunkSize property is a product of the MS-FSSHTP protocol used to both synchronize and store documents in SharePoint, and it represents a document as a set of "chunks" that can be incrementally synchronized or updated. As such, it mitigates the need to perform `0(FILE SIZE)` I/O on each change. Chunks or partition sizes are based on the structure of the source document and are selected to map as closely as possible to complete units of change or otherwise selected to align as closely as possible to the expected edits to a document.

BLOB Store Considerations (including Remote BLOB Storage)

Each read or write to a file involves reading and/or writing to a whole number of the file's partitions (there are not partial reads/writes) to the content database. There are two important characteristics to consider when storing BLOBs:

1. **Overhead**. A small overhead is presented when storing large quantities of smaller BLOB data, whereas a larger overhead is presented when storing fewer, larger BLOBs.
2. **Minimum Read/Write Granularity**. Minimum Read/Write granularity is the minimum amount of I/O that can be performed per input output processor. Read/Write of data less than this size will have the same cost as the full amount.

Individual BLOB storage and hardware characteristics have different values, this is more evident when you use Remote BLOB Storage providers. To manage these overheads and characteristics, Shredded Storage does not store file partitions directly in SQL Server. Conversely, a technique is employed that combines the partitions to form larger BLOBs of a given target size, specified through the FileWriteChunkSize property. The algorithm attempts to combine as many BLOBs as possible within the scope of an update operation to achieve the target size. However, that size is a target as opposed to an absolute and therefore cannot be accurately predicted by using an equation such as `FileWriteChunkSize / FILE SIZE = # Chunks`. In some scenarios, the algorithm will underfill BLOBs under certain conditions to optimize scenarios, and under other conditions the algorithm will overfill BLOBs when storing large data that does not require to be partitioned into more granular chunks. In addition, chunks are placed into a minimum of three (3) groups (dependent on chunk type) and the algorithm will not combine chunks across groups unless the file is under a specific size. As such it can be expected to have at least three (3) BLOBs for normal sized files (see also Storage Heuristics later in this paper).

The default FileWriteChunkSize is 64 KB and can be adjusted upward to 2 GB. The FileWriteChunkSize property in SharePoint 2013 manages the target size of shredded BLOBs on write operations and in addition the size of read in compound read-based operations.

WCF and SOAP Considerations

SharePoint Server 2013 implements WCF and Windows SOAP Stack components to ensure efficient and incremental Office document transfer over the wire. This transfer is based on a native-code implementation of SOAP that provides the core network communication functionality supporting a set of the WS-* and .NET-* family of protocols. This implementation is designed to facilitate co-authoring experiences that require an efficient file transfer in order to provide both the necessary performance and scalability to support such experiences. In this implementation, a client read is limited to a 3-MB partition of a given file. Therefore, a 9-MB file would be read in three compound operations (9 MB / 3 MB = 3). Now imagine a client opening the same 3-MB file where the FileWriteChunkSize property is configured to 2GB. It would pay an IO penalty on the storage subsystem of 2 GB for each 3 MB read by the client.

Reconfiguring the default FileWriteChunkSize (64KB) should be performed only in limited use cases, as this configuration can affect system-wide performance and availability. The value should be configured to be, at minimum, as large as the minimum Read/Write granularity of the storage. In addition, it should be large enough to achieve an acceptable overhead (based on the per-BLOB overhead of the store) without being too large that extraneous I/O occurs. Values of over 4MB should be avoided, as many operations consume changes in 4-MB increments and any amount over 4 MB will incur wasted I/O. For example, setting FileWriteChunkSize to 2 GB and downloading a 100MB file would incur 400MB of I/O as the file is accessed in 4MB increments that are 'rounded up' to the entire file size.

Configuring FileWriteChunkSize with Windows PowerShell

```
[void][System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SharePoint")
$service = [Microsoft.SharePoint.Administration.SPWebService]::ContentService
$service.FileWriteChunkSize = <size in b>
$service.Update()
```

Configuring FileWriteChunkSize with C#

```
namespace FileWriteChunkSize
{
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using Microsoft.SharePoint.Administration;

    class Program
    {
        static void Main(string[] args)
        {
            SPWebService webService = SPWebService.ContentService;
            webService.FileWriteChunkSize = 10240;
            webService.Update();
        }
    }
}
```

NOTE

A FileWriteChunkSize configuration cannot be used to disable Shredded Storage (i.e. 2GB). See also cautions in this white paper when configuring the FileWriteChunkSize property. Configuring the FileWriteChunkSize property should be thoroughly tested in a non-production environment.

Storage Heuristics

The SharePoint 2010 Cobalt implementation is an independent system built on top of a traditional full file representation of files stored in SharePoint 2010. Cobalt support is enabled through support of an additional BLOB of data. The file format metadata is stored next to the file and is used to describe the representation of the file. The file format metadata is used by the file data adapter to represent a read-only view of the file (see Figure 9).

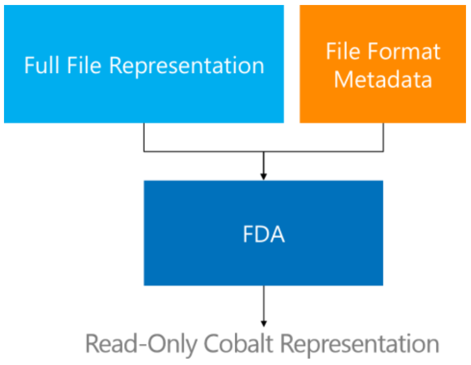


Figure 9 SharePoint 2010 Cobalt File Representation

The file format metadata contains a serialization of the full Cobalt data element graph with a complex reference scheme that allows the actual file data to remain in the full file representation.

To support write operations against the Cobalt representation, the file needs to be regenerated with a `O(FILE SIZE)` operation. In order to support higher frequency write rates, such as co-authoring, a temporary Read/Write store is used to delay file regeneration. This temporary Read/Write store is known as a hot table, and it logically augments the file format metadata state (see Figure 10).

Hot Table: Table in SQL Server used to store metadata and content related to a cold document used in coauthoring scenarios.

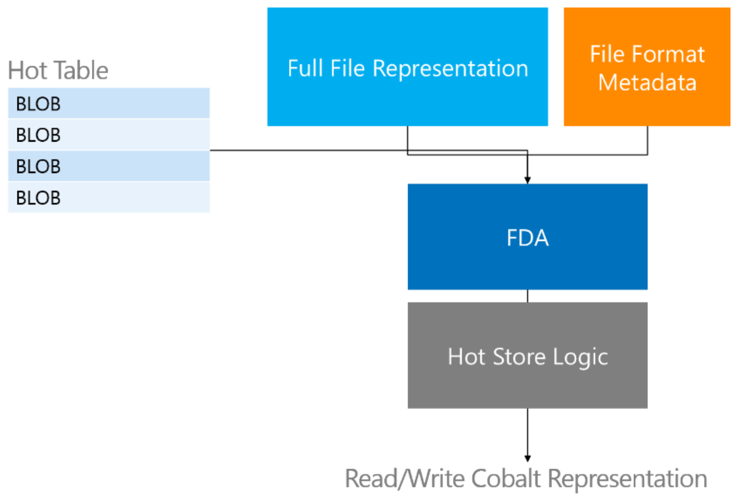


Figure 10 Hot Table Representation

In SharePoint 2013, the model is simplified by changing the representation to be natively shredded.

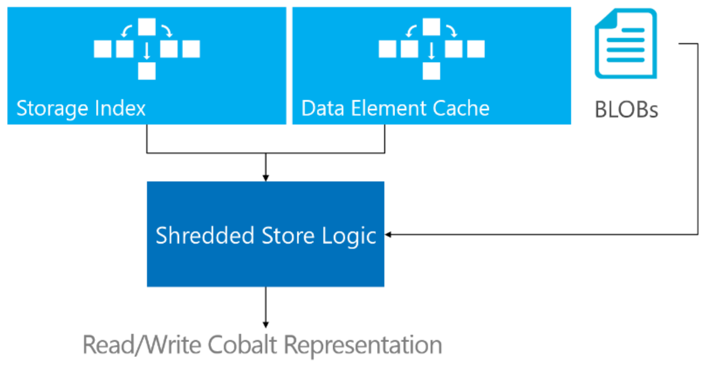


Figure 11 SharePoint 2013 Shredded Storage File Representation

SharePoint 2013 uses a new SPFileStream object to represent the state of a file-partition. This object exposes the interfaces for accessing the file-partition content as a traditional file stream or through Cobalt APIs. These APIs are not exposed or intended for public use.

By default, Shredded Storage partitions data across a 64 KB boundary based on the FileWriteChunkSize property, which is configurable up to a maximum partition size of 2 GB. 2 GB is also representative of the maximum file size supported by SharePoint 2013. However, a 2GB FileWriteChunkSize configuration would not result in a single BLOB as expected, but rather, at minimum, three entities as previously described in this paper. A master BLOB (which contains the indexes), a topology blob, and as many blobs are required for the leaf data.

Master BLOB

The master BLOB stores static high-level store information and specifically contains store version information, GUIDs used for knowledge generation, and the aggregate store's knowledge and data element hierarchy. It is written once (on the first PutChanges) and read, in its entirety, on every subsequent request.

Index BLOB

The index BLOB implements a mutable (key, value) pair store that contains the Cobalt storage index (i.e., storage manifest, cell manifest, and revision manifest mappings) and mappings for the master blob and data element hierarchy. Being a single blob, it is loaded, in its entirety, on every request and updated during PutChanges.

Data Element Cache (DEC) BLOB

The DEC implements a table containing an entry for every data element in the store. Each entry caches the type, size, and child references of the data elements. The DEC is used to efficiently traverse the references of the data element graph and reason about it. Being a single BLOB, it is loaded, in its entirety, on every request and updated during PutChanges.

Summary

Shredded Storage in SharePoint 2013 optimizes I/O and provides smoother I/O patterns when compared to prior versions of SharePoint Products, and it provides storage cost benefits under conditions where historical versioning is widely used and implemented.

Resources

SPWebService.FileReadChunkSize Property

SPWebService.FileWriteChunkSize Property

Appendix

Appendix A: Frequently Asked Questions

Question	**Answer**
Can Shredded Storage be disabled?	No, Shredded Storage is enabled by default and cannot be disabled.
Does Shredded Storage work with Remote BLOB Storage (RBS)?	Yes, Shredded Storage works with Remote BLOB Storage.
Can I prevent a file from being shredded?	No, Shredded Storage cannot be disabled.
Is BLOB data shredded when I upgrade to SharePoint 2013?	No. BLOB data is not partitioned until that data has been accessed and saved back to the server.
Are there any changes to IOP requires?	No. The published I/O recommendations for SharePoint 2010 are applicable to SharePoint 2013. For capacity planning information see also Capacity planning for SharePoint Server 2013.
Does Shredded Storage work with 3rd party RBS Providers?	Yes. Shredded Storage is compatible with 3rd-party RBS Providers.

Appendix B: Table of Figures

Appendix C: About the Author

Bill Baer is a Senior Product Marketing Manager and Microsoft Certified Master for SharePoint in the SharePoint Product Group at Microsoft Corporation in Redmond, Washington. Having previously worked at Hewlett-Packard, Bill Baer has a proven background in infrastructure engineering and enterprise deployments of SharePoint Products and Technologies.

While at Hewlett-Packard Bill Baer was awarded the MVP award for his contributions in the Technology Solutions Group, now known as HP Enterprise Business, which encompasses server and storage hardware, technology consulting, and software sales.

Blog: http://blogs.technet.com/b/wbaer