



4.0 Compound Document Model

| Table of Contents | |
|--|-----------|
| Overview. | 3 |
| IBM FileNet P8 4.0 Compound Document Model 3 | |
| <i>The Compound Document.</i> | 3 |
| <i>What Makes Up a Compound Document?</i> | 5 |
| <i>Shared Components.</i> | 5 |
| <i>Compound Document Relationship Types</i> | 6 |
| <i>Versioning</i> | 7 |
| Exploring a Compound Document Structure | 7 |
| Ordering Components in the Compound Document Structure. | 8 |
| <i>Server-based Component Ordering</i> | 8 |
| <i>Application-based Component Ordering</i> | 9 |
| Compound Document Security and Integrity | 9 |
| <i>Compound Document State Setting</i> | 9 |
| <i>Parent and Relationship Security</i> | 9 |
| <i>Cascade Delete Behavior</i> | 10 |
| <i>Prevent Delete Behavior</i> | 11 |
| Workplace User Interface | 12 |
| Development | 13 |

Overview

With the advent of various powerful editing and document composition tools that are able to create compound documents, many companies have realized the importance of providing enterprise-wide secure access to these documents and their component parts. A compound document is essentially a collection of components that are used together to form a single complete document.

Often also referred to as composite documents, this can be a very convenient and efficient way of creating a new document from existing components, enabling the easy re-use of intellectual property.

Using compound documents can provide benefits to the organization by allowing independent editing of various components, reuse of components in other documents, enhanced integrity during the publishing process, and savings in time. These capabilities allow large groups of users to collaborate on the creation of documents using a mix of applications, while assuring the integrity of the total documents as they pass through the process of creation, revision and distribution.

Different people using different applications may create portions of these documents at different times. Each portion of the document may have a different creation, approval and update cycle; yet the consumers of the document should see only a consistent result.

In our research we have found that each compound document solution views the way in which the components of the document are created, related, maintained, viewed and rendered in different ways. This has led IBM to develop the IBM FileNet P8 Compound Document model.

IBM FileNet P8 4.0 Compound Document Model

The goal of the FileNet P8 Compound Document model is to provide a framework that is extensible to encompass and facilitate many different compound document models such as DITA, DocBook, HTML, OLE, CAD, etc.

4.0 Compound Document Model

Page 3

Note: In the P8 4.0 release this is not a direct functional match with the Content Services Compound Document capabilities. Although the capabilities of the Content Engine will support Content Services Compound Document solutions, there are some out-of-the-box capabilities and User Interface components that currently do not have direct functional equivalents but may be added in the future.

The core P8 Compound Document capabilities are open, extensible and constructed in a way that many compound document models can be used simultaneously.

The core of any compound document model is to allow for the interconnection or linking of the various components of each document. For those who are familiar with IBM FileNet Content Manager you may be aware of many capabilities that exist today which support such interrelationships. Examples of these are Link objects, Object Value Properties and Referential Containment Relationships.

The goal of P8 4.0 is to formalize the Compound Document capabilities that exist today and to add capabilities which have been found to be common across all solutions. This formalizing of compound document capabilities is intended to provide our partners and customers a stable, supported foundation on which to implement their desired solutions.

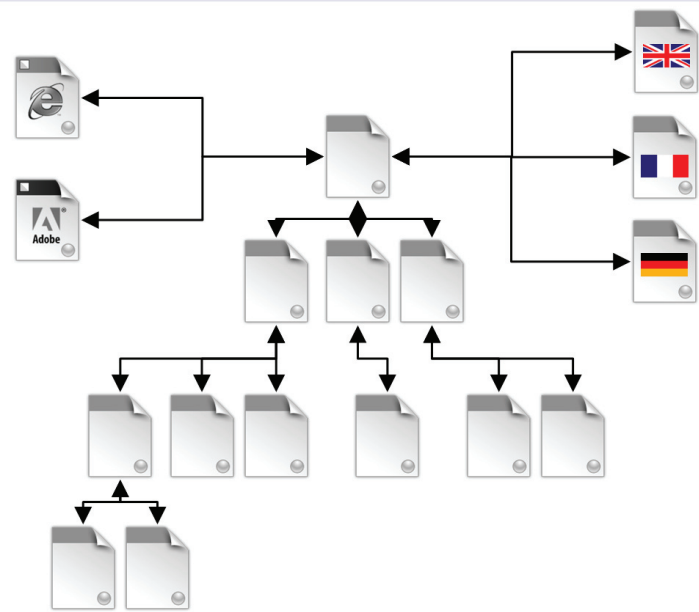
The Compound Document

In the P8 Compound Document model a parent document maintains relationships to immediate children. Those children in turn may be compound documents with children of their own. There is no anticipated limitation to the depth of the hierarchy.

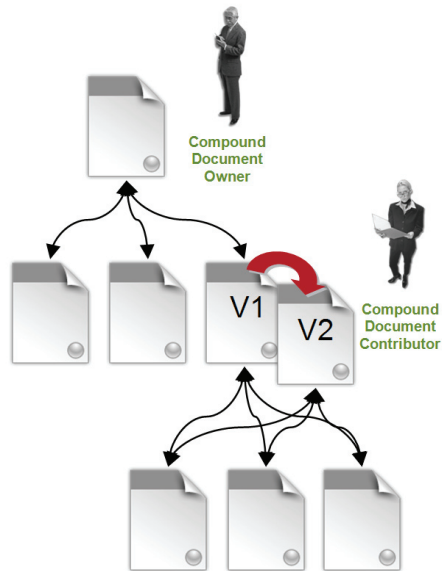
Although in many compound document models, the relationships indicate a hierarchal structure, this is by no means mandatory. The structure can also be articulated in a head-to-tail structure that indicates a relationship but not necessarily direct inclusion of the child document into the parent.

In the following diagram a single compound document is detailed as an example. Note that from the root parent document we have a hierarchy of child components with homogeneous data. In this example this constitutes the document itself. In addition, there are additional relationships to language translations and rendered versions of the assembled hierarchy. There are many other permutations and structures of compound documents that the Content Engine is now able to describe.

4.0 Compound Document Model



This model gives the largest degree of freedom for implementations. Child components may be complex compound documents in their own right with creation and review cycles following different ownership, creation and review cycles from the parent document.



4.0 Compound Document Model

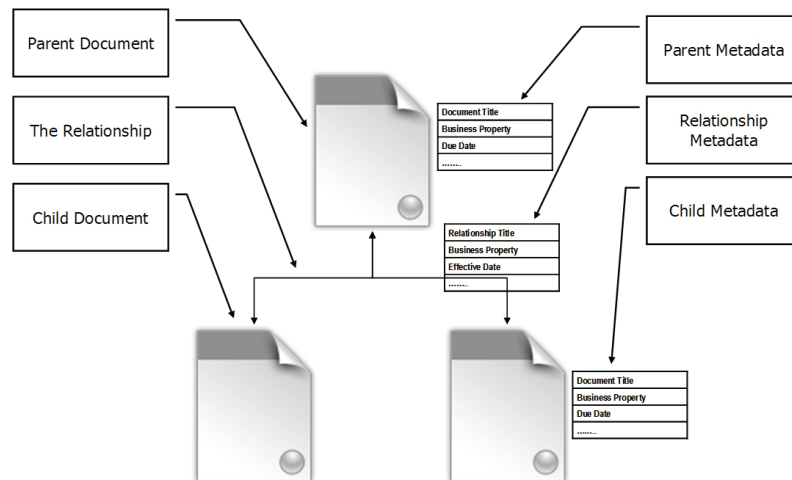
Page 5

What Makes Up a Compound Document?

The component parts of a compound document obviously include the parent and child documents in the structure. What they also include is the metadata on both and also on the relationship which connects them. If you are familiar with the Content Engine, you will be aware that document classes can have additional metadata properties assigned to them and that they can be sub-classed to derive additional classes from a similar parent. These properties are part of the compound document.

In addition to this, the compound document relationship is an object which can have additional properties assigned and can be sub-classed.

The result is a very rich structure for your application. A single compound document may have many different document types included and also have many different relationship types used to include child documents.

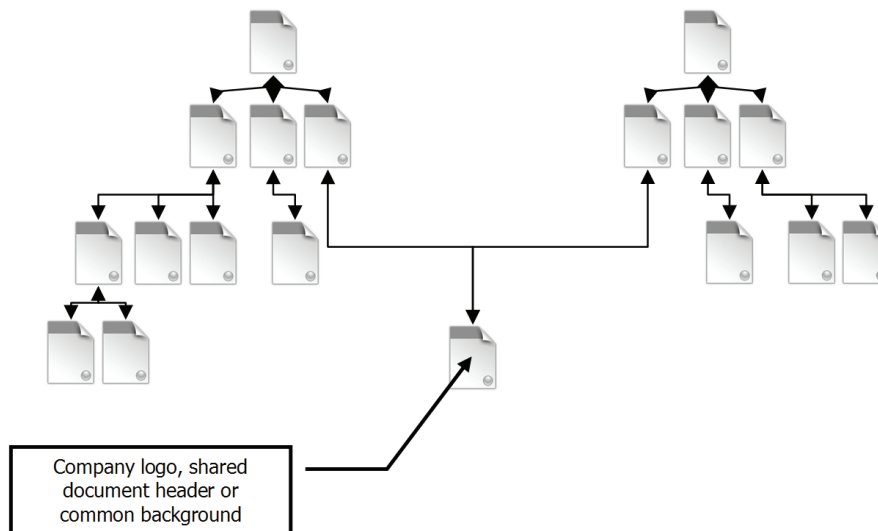


Shared Components

A single document or a compound document sub-tree may be used in many compound document structures as a child component simultaneously. This is commonly used in the cases of a company logo used in standard letters or a common title block on an engineering drawing. The benefit here is the consistency and control this enforces across all compound documents.

4.0 Compound Document Model

Page 6



Compound Document Relationship Types

A formal compound document relationship type is being introduced in Content Engine 4.0. The relationship will allow parent documents to be associated with the components which it is made up of. The components can be native content engine documents or versions, documents accessible via Content Federation Services and external document accessible via a URI. This allows an unprecedented level of inclusiveness for compound document authors in your organization.

In addition to the storage location of component documents, the relationship capabilities support the following types of associations:

| | |
|--|---|
| Link to static child component | Versioning of the child component does not affect the relationship. |
| Link to latest major version | The relationship always points to the latest major version of the child document |
| Link to latest version | The relationship points to the latest version of the child document regardless of the major/minor status. This is sometimes referred to as the "tip" version |
| Link to latest version with Label value set to X | A compound document label property is being introduced in all document classes. The child document will have such a label property. Each version of the child document could have this property set to a different value. The relationship will point to the latest version of the child component which has the label property set to a specified value. |
| Link to URL | A child component in this case is referred to by a URL such as a web URL. This enables the inclusion of components not under Content Engine control. |

Note: Relationships can be within the same object store and also across object stores. The primary qualification for cross object store relationships is that the object stores must be within the same Content Engine domain.

4.0 Compound Document Model

Page 7

The compound document relationship is designed to be sub-typed to suit your application document needs. This allows for type naming, custom property additions, default value setting, event triggering, auditing, etc.

Versioning

Another significant enhancement to support compound documents is the propagation of relationships during the versioning process of the parent document. When a compound document parent document has relationships with child documents and the parent document is versioned, these relationships are propagated to the next version.

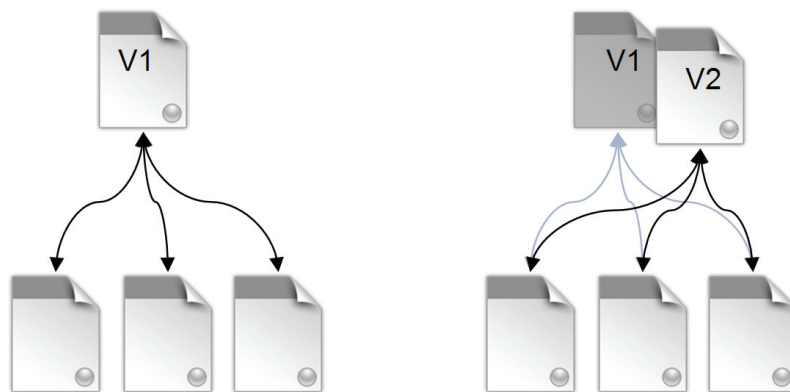


Figure 1 – Versioning a Compound Document Component

It is important to note that the prior version of the parent document is left intact during this process including its relationships. This means that any manipulation of the new version will not affect it.

Exploring a Compound Document Structure

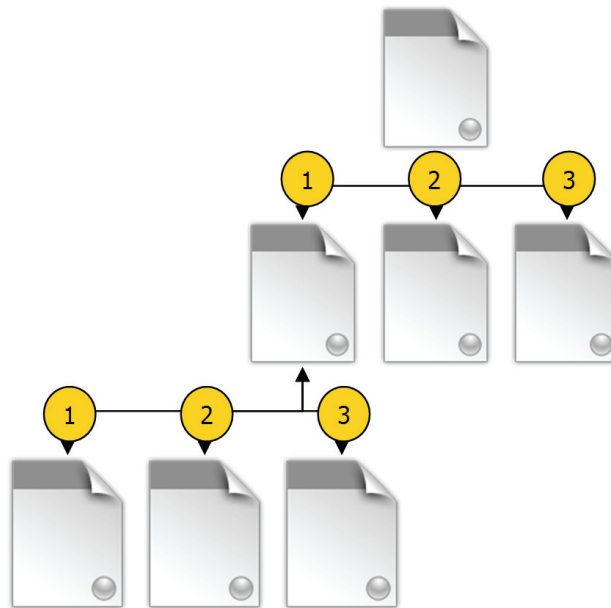
In order to allow applications and user interfaces to make use of a compound document, the Content Engine has extended the document object to allow for exploration of parent and child relationships. From any document in an object store the following can be retrieved.

| | |
|----------------------|---|
| Child relationships | This will retrieve any relationship objects that connect to compound document child objects of the current document. The relationship object will have a relationship type and may also have type specific metadata |
| Child documents | This will retrieve child document objects of the current document |
| Parent relationships | This will retrieve any relationship objects that connect to parent documents of the current document |
| Parent documents | This will retrieve parent documents of the current document |

Note: Since the parent document+relationship and child document are governed by independent security, it is possible that you will not be able to retrieve all relationships/documents if you do not have the required permissions.

Ordering Components in the Compound Document Structure

In many Compound Document solutions, the order of the relationships between components is a critical part of the document structure. The Content Engine ensures that relationships and component documents will always be presented in a reliable order. This order is maintained independently at each level of the structure.



Ordering can be provided in two different ways depending on the needs of the application.

Server-based Component Ordering

Given no additional input by the application, the Content Engine will manage a consistent ordering of the child components at each level of the hierarchy. If child relationships are created sequentially, this order will be maintained for all future actions.

The Content Engine APIs and the Workplace application both allow the components to be reordered.

4.0 Compound Document Model

Page 9

Application-based Component Ordering

Compound Document applications can choose to directly control the component ordering rather than defer to the content engine. The application can provide integer numbering for the components. The Content Engine will maintain and deliver the components based on this application defined ordering.

The server will not prevent collisions of sort order number values between components where the user supplies the sort order number.

Compound Document Security and Integrity

There are a number of capabilities and settings that allow compound document applications to maintain the integrity of their document structures.

Compound Document State Setting

In addition to security, the ability to create relationships to child components from a document is controlled by a document instance state setting named `CompoundDocumentState`. The two allowable states for this property are `idmStandardDocument` and `idmCompoundDocument`. If this property is not set to `idmCompoundDocument`, the document is considered to not be a compound document and therefore is not allowed to create relationships to child components.

The goal of this state setting is twofold. The first is to allow Workplace to represent whether a document is a compound document parent in result sets without incurring additional performance overhead. Workplace is also able to enable and disable compound document actions based on this state.

The second is to easily allow Compound Document applications to differentiate between documents that are compound in nature and therefore under their control.

This can be toggled between `StandardDocument` and `CompoundDocument`. A prerequisite for converting a `CompoundDocument` into a `StandardDocument` is that all child relationships must be removed before the Content Engine will allow this change.

4.0 Compound Document Model

Parent and Relationship Security

In a single compound document it is clear that the parent and child in any relationship have the ability to maintain separate security. The compound document relationship object, however, does not maintain a separate access control list. Access to existing relationship objects, and their creation and deletion is controlled by the parent object's access control list.

| | | |
|--|--|--|
| Create Relationship to Child Component | User must have FN_ACCESS_LINK rights on the parent component to create component relationship object that references the parent. User must have FN_ACCESS_READ rights on the child component to create component relationship object that references the child. | The Compound Document State property must already be set to CompoundDocument to permit adding a component relationship object. |
| Delete Relationship | Users must have FN_ACCESS_UNLINK rights on the parent component to delete component relationship object that references the parent. | Any user with FN_ACCESS_UNLINK rights on the parent component is also granted DELETE rights on any direct child component relationships object. |
| Update Relationship Properties | Users must have FN_ACCESS_WRITE rights on the parent component to modify properties on a component relationship object. | Property update rights allow users to modify order number, change child document referenced, and modify custom properties on component relationship objects. |

Cascade Delete Behavior

The compound document model provides a capability to further solidify the relationship between a parent and a child component. This is provided as a relationship property titled Cascade Delete. If this is set for a relationship, when the parent is requested to be deleted, the child component referred to by this relationship will also be deleted. This is appropriate for compound document solutions where all components in the structure are considered to be the same document. Having sufficient access to all documents in the structure that are connected with relationships set to cascade delete is required. If you do not have the correct level of access, you will not be allowed to complete the action.

4.0 Compound Document Model

Prevent Delete Behavior

In order to further control and maintain compound document integrity a relationship property called ComponentPreventDelete can be used.

| | |
|-----------------------------|--|
| ALLOW_BOTH_DELETE (default) | No control on delete behavior |
| PREVENT_CHILD_DELETE | <p>This prevents a child component from being deleted if it is included in an compound document with a relationship set with this value.</p> <p>When an independent author is working with a child component and is unaware of the compound document hierarchy it is included in, this will prevent the deletion of the document if its existence is critical to the integrity of the document as a whole.</p> <p>Both an individual deletion request on the child and also a cascade delete that involves that child will be prevented.</p> |
| PREVENT_PARENT_DELETE | <p>This prevents the parent being deleted if any of the relationships to child components are set to this value.</p> <p>This could be used ensure that a parent document would not be deleted if there were child components which would be considered orphaned if it were to be removed.</p> |
| PREVENT_BOTH_DELETE | Neither the parent nor the child can be separately deleted. |

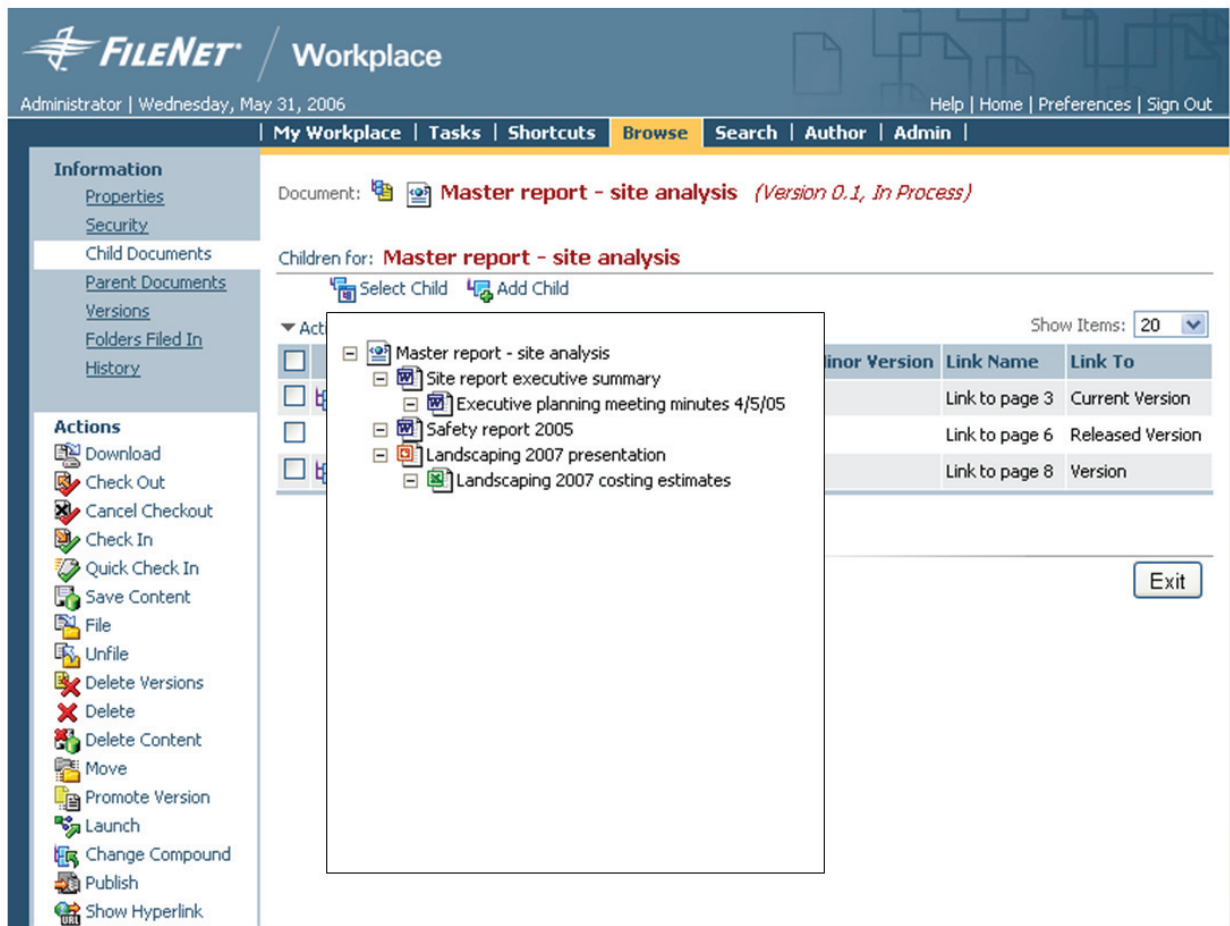
4.0 Compound Document Model

Page 12

Workplace User Interface

The Workplace application has been extended to expose the compound document model extensions to the user. This allows for the creation, modification and deletion of relationships. It also allows for the exploration of parent and child relationships from any document.

The capabilities are generic to the compound document framework. The user interface does not, for example, implement Microsoft OLE Compound Document capabilities. This would require additional development at this time.



The screenshot displays the FILENET Workplace interface. At the top, the header includes the FILENET logo, the word "Workplace", and navigation links: "Administrator | Wednesday, May 31, 2006" and "Help | Home | Preferences | Sign Out". Below the header is a navigation bar with tabs: "My Workplace", "Tasks", "Shortcuts", "Browse" (selected), "Search", "Author", and "Admin".

The main content area shows a document titled "Master report - site analysis (Version 0.1, In Process)". Below the title, it lists "Children for: Master report - site analysis" with options to "Select Child" and "Add Child". A tree view shows the following structure:

- Master report - site analysis
 - Site report executive summary
 - Executive planning meeting minutes 4/5/05
 - Safety report 2005
 - Landscaping 2007 presentation
 - Landscaping 2007 costing estimates

To the right of the tree view is a table with columns "Minor Version", "Link Name", and "Link To". The table contains three rows of data:

| Minor Version | Link Name | Link To |
|---------------|----------------|------------------|
| | Link to page 3 | Current Version |
| | Link to page 6 | Released Version |
| | Link to page 8 | Version |

At the bottom right of the main content area is an "Exit" button. On the left side, there is a sidebar with "Information" (Properties, Security, Child Documents, Parent Documents, Versions, Folders Filed In, History) and "Actions" (Download, Check Out, Cancel Checkout, Check In, Quick Check In, Save Content, File, Unfile, Delete Versions, Delete, Delete Content, Move, Promote Version, Launch, Change Compound, Publish, Show Hyperlink).



Development

There are a number of development API languages that can be used to develop applications against the Content Engine. Compound document capabilities are exposed in these APIs

CE 4.0 JAVA API; CE 4.0 .Net API; CE Web Services API

The following APIs have partial or no exposure of the compound document.

CE 3.5 JAVA API – It is possible to discover if a document is a compound document parent. Compound document method extensions and relationship objects are not exposed. In a single application it is possible to combine CE 3.5 and 4.0 JAVA API code to form a single solution.

CE COM APIs – It is possible to discover if a document is a compound document parent. Compound document method extensions and relationship objects are not exposed.

© Copyright IBM Corporation 2007

IBM Corporation
3565 Harbor Boulevard
Costa Mesa, CA 92626-1420
USA

Printed in the USA

07-07

All Rights Reserved.

IBM and the IBM logo are trademarks of IBM Corporation in the United States, other countries or both. All other company or product names are registered trademarks or trademarks of their respective companies.

For more information, visit
ibm.com/software/data/cm.