UCCX Repository Restrictions

Document ID: 116496

Contributed by Ryan LaFountain and Siddachetty Mahesh, Cisco TAC Engineers.
Sep 12, 2013

Contents

Introduction

This document describes an issue encountered when you upgrade Cisco Unified Contact Center Express (UCCX) systems to Versions 8 and later and a large number of Repository items are uploaded to the system, or when you attempt to upload a large number of Repository items to the system in Versions 8 and later.

Background Information

UCCX Versions 7.x and later use Microsoft SQL (MSSQL) as the database engine. MSSQL does not distinguish, in terms of data storage, between data of different types. When it stores data in a 3−GB database, MSSQL stores all data, regardless of type, in one 3−GB block.

By contrast, Informix, the database engine used in UCCX Versions 8.0 and later, differentiates between data of different types when it stores it on the disk. Typical database data (such as Strings, Characters, and Integers.) are stored in a disk chunk devoted to the database, whereas Binary Large Object (BLOB) data, if any exists in the database table records, is stored in a separate section of the disk, called an sbspace. An sbspace is a logical unit composed of one or more disk chunks that store BLOB data. Informix stores traditional data and BLOB data separately in order to increase the performance of reading and writing BLOB data from and to the database and disk. When a database is created that contains BLOB data, the administrator must specify the size of the disk chunks for the database (in order to store traditional data) and the size of the sbspace separately.

For data storage mechanisms, MSSQL places all data into a single bucket of size N, while Informix divides the storage of this data into two buckets: one bucket for the contextual information about BLOB data of size X, and another bucket for the BLOB objects themselves of size Y.

In UCCX, the administrator has the option to upload Repository items that consist of Prompts, Documents, Grammars, and Scripts. The contents of these items are stored in the corresponding database tables as BLOB data and contextual information about them, such as the filename, folder, last modified time, last modified user, length, and checksum.

Repository items are stored in the UCCX database ***db_cra_repository***. In UCCX Versions 7.x and earlier that use MSSQL, the ***db_cra_repository*** is 3 GB in size and contains both the contextual and BLOB information. In UCCX Versions 8.0 and later that use Informix, the data storage chunk attached to the ***db_cra_repository*** is 10.2 MB in size, and only stores the contextual information about the Repository items. The content of the Repository items is stored in BLOB format in a sbspace called ***uccx_sbspace***. In UCCX Versions 8.0 and later, the ***uccx_sbspace*** is 3 GB in size.

The output of ***show uccx dbserver disk*** on a UCCX Version 8.0+ server, reveals the distinction between these two datastores:

```
admin:show uccx dbserver disk
SNO. DATABASE NAME      TOTAL SIZE (MB) USED SIZE (MB) FREE SIZE (MB) PERCENT FREE
---- ------------------ --------------- -------------- -------------- ------------
 1    rootdbs                    358.4           59.5          298.9          83%
 2    log_dbs                    317.4          307.3           10.1           3%
 3    db_cra                     512.0           17.3          494.7          96%
 4    db_hist                  34508.6         6661.3        27847.3          80%
 5    db_cra_repository           10.2            3.4            6.9          67%
 6    db_ifxcat                  512.0            3.3          508.7          99%
 7    temp_uccx                 1572.9            0.1         1572.7          99%
 8    uccx_sbspace              3145.7         2988.1          157.6           5%
 9    uccx_ef                    204.8            0.1          204.7          99%
 10   uccx_ersb                 1572.9         1494.1           78.8           5%
 11   sadmin                     102.4            4.3           98.1          95%
```

Depending on the mix of data in the MSSQL database, it is possible for the size of the BLOB data stored in the MSSQL database to exceed the defined size of the sbspace in Informix when migration or upgrade is attempted. Similarly, it is possible that the contextual information about the BLOB data stored in the MSSQL database exceeds the administratively−specified size for that data in the Informix database chunk.

When this occurs, upgrade or migration from UCCX Version 7.x to UCCX Version 8.x fails, because either the ***db_cra_repository*** or the ***uccx_sbspace*** is not large enough to accomodate the same information that was stored in MSSQL. This is typically an issue in an UCCX system that contains a large number of Prompts. Contextual Prompt and BLOB data must share the ***db_cra_repository*** and the ***uccx_sbspace*** with Documents, Grammars, and Scripts, but these other Repository types are typically small in size and number.

As an example, consider a UCCX Version 7.x system with tens of thousands of Prompts, each with only a few seconds of audio. In UCCX Version 7.x that uses MSSQL, the Prompt content and contextual information is stored in the same 3−GB chunk. As there are many Prompts of small size, the database might store 50 MB of contextual information about the Prompts, but only 2 GB of BLOB data that represents the audio of the Prompts. Therefore, the Prompts in the Repository occupy a little over 2 GB of the 3−GB limit set upon database creation.

When you attempt to migrate this system to UCCX Version 8.x and Informix, the migration fails because the 50 MB of contextual information exceeds the 10.2 MB limit of the ***db_cra_repostiory*** even though the 2 GB of Prompt content fits well under the limit of the ***uccx_sbspace***.

Inversely, consider a UCCX Version 7.x system with fewer, but still many, long Prompts. With fewer Prompts but of larger size, the ratio of Prompt content to contextual information is different. In UCCX Version 7.x and MSSQL, the Prompt content might take up 2.8 GB of the ***db_cra_repository***, and the contextual information 3 MB. This system upgrades successfully, as the 3 MB fits into the ***db_cra_repository***, and the 2.8 GB fits into the ***uccx_sbspace*** allocated.

Typically, when you attempt to migrate to UCCX Versions 8.x and later, the contextual data about the Prompts uploaded to the UCCX Versions 7.x or earlier system exceed the size limit of ***db_cra_repository*** before the Prompt content exceeds the size limit of the ***uccx_sbspace***. Additionally, the true free space available for customized Repository items is 6.9 MB, as the default configuration consumes 3.4 MB of the

db_cra_repository.

Problem

When you attempt to upload new Repository items (Documents, Grammars, Prompts, Scripts) to a UCCX system that runs Versions 8 or later, you receive this error message:

```
The files uploaded are not valid or not structured
 according to languages. Please check the help
 documentation for more details.
```

The migration from UCCX Versions 7.0(2) and earlier to Versions 8.0 and later changes the operating system and database engine on which the application runs. The database engine used in UCCX Versions 8.0 and later stores data differently than that of UCCX Versions 7.x and later. This has implications for the migration of UCCX, as databases that contain large datasets in UCCX Version 7.x might not migrate properly to UCCX Version 8.x.

Solution

Before you migrate to UCCX Version 8.x, estimate the amount of ***db_cra_repository*** and ***uccx_sbspace*** needed in order to store the current Repository items in the UCCX Version 7.x system, to include any future growth.

Estimate Current Repository Data Size

In order to begin, determine the number of rows in each of the Repository tables that hold information on both Repository items and folders.

Determine Number of Rows in Repository Folder Tables

Use Microsoft SQL Query Analyzer in order to record the number of rows from the Repository folder tables with these commands:

- ***SELECT COUNT(*) FROM documentsfoldertbl***
- ***SELECT COUNT(*) FROM grammarsfoldertbl***
- ***SELECT COUNT(*) FROM promptsfoldertbl***
- ***SELECT COUNT(*) FROM scriptsfoldertbl***

Determine Number of Pages used by Repository Folder Tables

Informix accounts for size−on−disk in terms of pages. Determine the number of pages that is occupied by the content of the Repository folder tables with this formula, and substitute the ***Number of Rows*** for the counts obtained from the commands previously−mentioned. Compute this formula for each table, and add the number of pages. It is not possible to accurately determine the number of pages if the number of rows from each table is added first, and then the formula result is computed.

- # Pages documentsfoldertbl = ***Number of Rows in documentsfoldertbl*** / (2020 / (180 + 4))
- # Pages grammarsfoldertbl = ***Number of Rows in grammarsfoldertbl*** / (2020 / (180 + 4))
- # Pages promptsfoldertbl = ***Number of Rows in promptsfoldertbl*** / (2020 / (180 + 4))
- # Pages scriptsfoldertbl = ***Number of Rows in scriptsfoldertbl*** / (2020 / (180 + 4))

Pages documentsfoldertbl + # Pages grammarsfoldertbl + # Pages promptsfoldertbl + # Pages scriptsfoldertbl = ***Total Number of Pages for Folder Tables***

Determine Number of Pages used by Repository Item File Tables

Complete the same calculation in order to determine the total number of pages for the file tables that contain the actual Repository items. Enter these commands with Microsoft SQL Query Analyzer:

- ***SELECT COUNT(*) FROM documentsfiletbl***
- ***SELECT COUNT(*) FROM grammarsfiletbl***
- ***SELECT COUNT(*) FROM promptsfiletbl***
- ***SELECT COUNT(*) FROM scriptsfiletbl***

Determine the number of pages that are occupied by the content of Repository file tables with this formula, and substitute the ***Number of Rows*** with the counts obtained from the commands previously−mentioned. Compute the formula for each table, and add the number of pages.

- # Pages documentsfiletbl = ***Number of Rows in documentsfiletbl*** / (2020 / (229 + 4))
- # Pages grammarsfiletbl = ***Number of Rows in grammarsfiletbl*** / (2020 / (229 + 4))
- # Pages promptsfiletbl = ***Number of Rows in promptsfiletbl*** / (2020 / (229 + 4))
- # Pages scriptsfiletbl = ***Number of Rows in scriptsfiletbl*** / (2020 / (229 + 4))

Pages documentsfiletbl + # Pages grammarsfiletbl + # Pages promptsfiletbl + # Pages scriptsfiletbl **= *Total Number of Pages for File Tables***

Calculate

Perform these calculations in order to finish the current Repository data size estimation:

1. Determine the total number of pages necessary in order to store the current Repository in Informix with:

   Total Number of Pages = Total Number of Pages for File Tables + Total Number of Pages for Folder Tables

2. Determine the total space, in MB, that the pages consume:

   Total Number of Pages x 2 = Total Size in MB

If the calculations show that the contextual information about the Repository items and folders currently uploaded in UCCX Version 7.x exceeds 3.4 MB, then it is recommended to refactor the Repository item design. Although the available free space for the contextual information about Repository items in ***db_cra_repository*** is 6.9 MB, it is recommended to leave 50% available for future growth. The growth estimates and maximum allowable occupied space is calculated per deployment, based on expected growth factors.

Prompts

Since Prompts are typically the largest consumer of the Repository space, methods used in order to reduce the number of Prompts in the Repository are discussed in the rest of this article.

If the Prompts currently uploaded in the UCCX Version 7.x Repository occupy a significant portion of the overall Repository storage space, refactor the Prompt design, storage, and retrieval before you migrate to UCCX Version 8.x. When you attempt to refactor the Prompt design, consider these options:

- Reduce the number of Prompt folders. As shown in the previous calculations, each Prompt folder occupies a row in the Repository database, and therefore consumes space against the

db_cra_repository limit. If you reduce the number of folders, you can free space for files.

- Share Prompts for common words or phrases across Applications/Scripts in order to reduce the number of Prompts needed in the system.
- Use the system−provided Prompts for commonalities among Prompts, such as numbers and currencies.
- Store Prompts on a separate web server within the enterprise, and retrieve and play the necessary Prompts on demand with Voice XML (VXML).

VXML is used in order to retrieve and play Prompts on−demand from an off−box location. If you store large amounts of Prompts on a separate web server, you can:

- Eliminate the migration problems previously mentioned, as Prompts are no longer stored in the ***db_cra_repository***.
- Provide greater access and ease of management.
- Reduce switch−version, upgrade, migration, and backup time.

Although many options exist in order to instrument Interactive Voice Response (IVR) customization in VXML, the UCCX script and VXML application used in order to retrieve a Prompt from an off−box web server and play it to the caller is used as a basis for further development. Much like other custom scripting in UCCX, the scripts provided in this section are offered as a guide and are not supported by the Cisco Technical Assistance Center (TAC).

Note: A VXML application is invoked from a UCCX script with the ***Voice Browser*** step. More information about the ***Voice Browser*** step is provided in the Cisco Unified Contact Center Express Programming Guides.

The ***Voice Browser*** step consumes a VXML document. This document must be created as the result of the ***Create URL Document*** step, and must be hosted on a webserver external to UCCX. Although the VXML application is written in order to accept caller input through Dual Tone Multi Frequency (DTMF), this application is designed only to play a prompt that is hosted off−box. However, it can be expanded in order to include additional functionality. It is assumed that the rest of the UCCX script, before the ***Voice Browser*** step is invoked, has the logic needed in order to determine which prompt is played, and a String variable set to the Prompt filename.

Since the VXML document is static, but the Prompt played through it is dynamic, a server−side scripting language is used in order to create the VXML document. This can be any server−side scripting language that has the ability to set the Content−type header of the XML ***GET Request*** response. In this example, PHP is used.

The PHP page is written in order to accept a URL parameter in a ***GET Request*** that represents the audio Prompt name that is played. The PHP page concatenates the VXML template with the Prompt filename passed in the ***GET Request*** URL parameters in order to form the complete VXML document. It then sets the Content−type header of the response to XML, and sets the body of the response to be the VXML content.

```
<?php
$wav_filename = $_GET['wav'];

$xml_string = '<?xml version="1.0"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.0">
   <form>
       <block>
           <prompt bargein="true">
               <audio src="http://<Servername or IP Address>/
               <Path>/'.$wav_filename.'.wav" />
           </prompt>
       </block>
   </form>
</vxml>';
```

```
header('Content-type: text/xml');
echo $xml_string;
?>
```

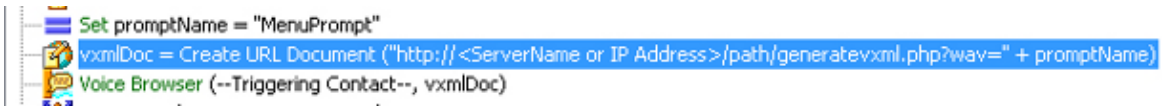
In order to produce a well−formed VXML document, the example PHP page must be accessed with a ***GET Request*** that contains a parameter ***wav*** and a String value, with the assumption that the example PHP page is named ***generatevxml.php***:

http://<Servername or IP Address>/path/generatevxml.php?wav=MenuPrompt

Ensure that the ***MenuPrompt.wav*** is in the location on the external webserver specified in the VXML template contained within the PHP page.

In the UCCX script, use the ***Create URL Document*** step in order to perform a ***GET Request*** of ***generatevxml.php*** concatenating the base URL of ***http://<Servername or IP Address>/path/generatevxml.php?wav=*** with the Prompt filename derived from the previous scripting logic, and place the result in a document variable.

Create a ***Voice Browser*** step that consumes the document variable.



When this script is called, provided both the ***generatevxml.php*** and ***MenuPrompt.wav*** are accessible on the webserver from UCCX, the ***MenuPrompt.wav*** Prompt plays to the caller.

When VXML applications are used in order to store Prompts off−box, so that they are accessed only when needed in order play them to the caller, it allows for greater efficiency, manageability, and serviceability. This is an issue for consideration if a UCCX Version 7.x system is upgraded to a UCCX Version 8.x system, and the number of Prompts are such that the content of the contextual information is larger than the ***db_cra_repository*** or the ***uccx_sbspace***.

Updated: Sep 12, 2013 Document ID: 116496
