TELESCOPE DATABASE INTERNALS GUIDE

VERSION 9.4.0.17

February 12, 2019

northplains™
Connecting your world. Visually.

Table of Contents

Overview

The core of the Telescope application is its database, which stores the majority of the application data, including asset information or metadata. Telescope supports prominent database vendors, operating systems, and hardware. Databases such as Oracle, and Microsoft SQL Server, and systems such as Sun Solaris, HP, Windows (Intel), and Linux (Intel) are commonly used with Telescope implementations. Consequently, the Telescope application and database have a "vanilla" design, with few vendor-specific features that could impair cross-platform or cross-database compatibility. As a result, you must make minor adjustments to the Telescope database to meet your unique performance and functional requirements. To meet this need, the product is built to be extremely flexible and powerful in the hands of a knowledgeable administrator.

The purpose of this guide is to help Telescope administrators:

- Understand the data structure of the Telescope database
- Achieve a working knowledge of the relationship between the database tables
- Customize the Telescope environment using stored database code (triggers, procedures, and functions)
- Tune the database to maximize Telescope performance in their environments

This guide lists the tables and their columns, explains their uses, and provides tips for using them. It also includes information about using triggers, procedures, and functions, and making the best use of the Telescope query generator. Finally, exercises (and their possible solutions) help you apply the information to real world scenarios.

Triggers, Stored Procedures, and Functions

The Telescope database uses triggers, stored procedures, and functions to enhance its functionality. You can use these procedures and functions to develop custom functionality, but since Telescope depends on them, you must not alter them in any way.

WARNING: Do not delete, disable, or modify any Telescope trigger, procedure, or function.

Telescope triggers are described in the table description where they are used. Telescope procedures and functions are described in Appendix B.

Database Tables

The Telescope application uses tables that can be categorized according to nine functions:

Metadata tables store information about the Telescope metadata schema, and assist with data entry. These tables provide data validation, popup menus, and cascading fields.

Asset Data tables contain all the information about assets. The information in these tables ranges from customized fields and keywords to file size and modification dates of individual assets.

Functional Rule tables store information about the Telescope functional rule scripting capabilities.

Order Entry tables manage and store data about the structure of the order entry system, and the orders placed by users.

User tables contain information about Telescope users. Personal settings in the Telescope application are stored in these tables, along with the Telescope user permission model.

Catalog tables store information about Telescope collections (catalogs).

Messaging tables store Telescope messages between users, including attachment of assets.

Search tables store information about various different kinds of Telescope searches that are available to users to easily locate asset records in the database.

System tables store application-wide information. Data such as application settings, user statistics information, and even installed customizations are stored in these tables.

Feature-specific tables store information required by certain Telescope features, such as Order Processing and Orchestration.

Note: In the descriptions of data types for these tables, the nvarchar(max) type is the MS SQL data type; Oracle equivalents may be either CLOB or NCLOB.

Metadata Tables

The following tables handle the structure of Telescope metadata and its validation:

- CASCADE_FIELDS
- EXTRA_COLUMNS
- ICONIC_FIELDS
- POPUPS

These tables are described in detail in the sections that follow.

CASCADE_FIELDS

Cascading fields are metadata fields that are visible or hidden in Telescope clients depending on values of another "parent" field. For example, imagine an environment with the following metadata fields:

Asset Type: a popup menu with a choice of "Project" or "Document"

Project Name: a free form field for the name of the project

Document Description: a free form field for the description of the document

If the user picks an Asset Type of "Project," the field "Project Name" should become visible and "Document Description" should be hidden. If the user picks an asset type of "Document," the field "Document Description" should become visible and "Project Name" should be hidden.

The following chart describes the columns in the CASCADE_FIELDS table.

Field	Data Type	Description
column_idx	short integer	ID of the field whose cascade properties are being described. It is a reference to the ID column in the *EXTRA_COLUMNS* table. In other words, this is the "child" ID.
cascade_column	short integer	ID of the field that controls the visibility of the field being described. It is also a reference to the ID column in the *EXTRA_COLUMNS* table. In other words, this is the "parent" field ID.
cascade_value	short integer	This is the value of cascade_column for which the field being described should be visible. It is an index into the popup menu for the parent field and thus a reference to the popup_idx column in the *POPUPS* table.

COLUMN_DISPLAY

The following chart describes the columns in the COLUMN_DISPLAY table, which is an additional lookup table for non-translatable fields to have localized labels (display names).

Field	Data Type	Description
column_id	short integer	Cross-reference to the column ID in the EXTRA_COLUMNS table.
display_name	nchar(100)	The localized name displayed for the column corresponding to the lang_id language. For example, en_US.
lang_id	nchar(10)	The language of the display name. Display names will be displayed if this setting matches the user local setting.

EXTRA_COLUMNS

The EXTRA_COLUMNS table is the "data dictionary" that Telescope uses to define every metadata column used in the Telescope environment. It is populated when administrators add metadata fields through the TSAdmin interface. (Manual database updates are not recommended.) Columns that are not listed in the EXTRA_COLUMNS table will not appear as metadata fields in the Telescope environment.

Columns containing metadata information are added to the EDITORIAL table by default, but some customer environments may need to split the data across several tables, for example if there is too much data to be contained in the one EDITORIAL table, or if there are external tables that need to be kept separate. Normalized repeating fields also require separate tables to contain their data. If you need to add these external tables, follow the instructions below.

Adding External Tables

If you need to use tables outside of the EDITORIAL table, follow these instructions:

1. Use the tsp_createTableExtendEditorial procedure to manually create the external tables. We strongly recommend using this procedure to ensure all verifications are performed and the mandatory record_id field and all keys and constaints are included. This procedure will also set up a foreign key relationship to the editorial table so that the new table is visible in TSAdmin.
2. Manually add all required columns to this new table. Use names that will identify the metadata fields you will be creating. Note that only one additional column is required if you are creating a table for Normalized Repeating fields.

3. Use TSAdmin (Fields tab) to add each of the new fields. Select the new table from the Table Name pulldown (it will be available there if you used the above procedure) and then select the appropriate column from the Field Name pull-down. (For normalized repeating fields, only one field is available.)
4. Specify the rest of the information as required. The information you complete in this Metadata field section is the data stored in the EXTRA_COLUMNS table.

Note: There are a few minor limitations on metadata that is stored externally (outside of the EDITORIAL table). These limitations include the inability to perform hierarchical searches on these columns and the ability to have these columns appear in the thumbnail view of the application. Future releases of Telescope will not have these limitations.

EXTRA _COLUMNS Fields

The following chart describes the columns in the EXTRA_COLUMNS table.

Field	Data Type	Description
id	short integer	Unique ID representing the column; referenced by other tables in the Telescope schema to uniquely link data back to a column.
column_name	nchar(32)	Name of the column as it appears in the database table.
viewer_name	nchar(32)	Default name of the column, as users will see it in the Telescope application.
table_name	nchar(32)	Name of the table where the column resides. In most cases, this field will contain "editorial," because the majority of columns defined in Telescope reside in this table. However, Telescope supports fields in extra_columns that do not exist in the *EDITORIAL* table, both for display and editing purposes. Note:, Normalized Repeating fields also use external tables. .
data_type	integer	A number that specifies the type of data contained in the metadata field's column created in the EDITORIAL table. **1 Char** (up to 255 characters) **2 Longchar** (over 255 characters). Some DBMSs (like Oracle) define a maximum length on this type of field and others do not. The max_len field can be used to limit the number of characters allowed in this type of field for a particular DBMS. **3 Integer** (4-byte integer) **4 Short Integer** (2-byte integer) **5 TimeStamp** could contain the date, time, or both **6 Boolean**. This is not a Boolean field, but actually a one-character field with the value "Y" indicating true. any other

Field	Data Type	Description
		value (except NULL) indicating false, and NULL indicating not specified. The NULL value is only allowed if the field is not required (required_yn below cannot be "Y"). **7 Repeating fields** are a delimited list of text stored as a longchar field. Telescope displays the values as a list, parsing out the delimiter, which is a vertical pipe character ("|"). The maximum number of entries in this list is limited by the length of the field (max of 2000 characters). An example of a list would be: "|this|is|an|example|." **8 Normalized Repeating** fields are a list of values stored in a separate, normalized table. It is treated, for display purposes in Telescope as a standard repeating field. The individual values are retrieved from separate records in an external normalized table (which needs to be created externally). These fields are recommended over Repeating fields. **9 Real**. This is represented in the database as a floating-point number in the database's internal format. **12 Container Field – number**. This field is a counter that indicates the number of child assets in the container field. A parent asset is linked to a child asset by entries in the *DOC_LINKAGES* table. (For details, see the explanation for that table.). 14 Iconic Field. 15 Separator. **99 I-Piece defined**. This is actually a Longchar field that is handed by an I-Piece in the Telescope structure. The contents of this field are known only by the I-Piece that controls the field, and Telescope treats the field as an opaque structure. (This field is relevant to Version 8.5, not to 9.x.)
max_len	integer	Maximum number of characters that can be put into the field. This is only used for Char, Longchar, timestamp, and Repeating fields (i.e., fields whose data_type value is 1, 2, 5 or 7). For timestamp fields, the value of this field determines how the information is displayed. A value of 1 indicates Date, 2 indicates Time, and 3 indicates Both. For Real fields (i.e., data_type 9), this field gives the number of decimal places that should be displayed. For other data types, this field is ignored.
validate_yn	nchar(1)	"Y" in this field indicates that the field is validated against its popup menu before the user is allowed to continue. Any other value (including NULL) in this field indicates that the field can contain any value, and the popup menu for the field (if there is one) is used as a data-entry tool only. TsWeb highlights required fields with an asterisk and/or different color.

Field	Data Type	Description
required_yn	nchar(1)	"Y" in this field indicates that the field must contain a value before the user is allowed to continue. Any other value (including NULL) in this field indicates that the field may be left blank.
custom_yn	nchar(1)	"Y" in this field indicates that the user can add values to the popup menu. These values are visible only to the user who added them. This field is not recommended for controlled vocabulary because only the defining user can see it. Use instead the POPUPS table.
priv_lvl	integer	DEPRECATED.
colcascade_yn	nchar(1)	"Y" in this field indicates that the field's visibility depends on the value of another field.
lookup_yn	nchar(1)	"Y" in this field indicates that this field is a lookup-enabled field facilitated by the Lookup Broker. This field and the lookup.xml file must match. For example, for ISBNs; all occurrences must be synched across your environment.
col_properties	nvarchar(4000)	The properties values for the Telescope field. This text field contains a set of property values in standard Windows '.ini' attribute format (i.e. name = value), with each attribute/value pair separated by a newline (ASCII 10 or ASCII 13 or both).
distribute_yn	nchar(1)	"Y" in this field will mark the field to be distributed by the Distribution Broker. When set to "N", the field is ignored by the Distribution Broker.
searchon_yn	nchar(1)	"Y" in this field will mark the field to be included in searches. When set to "N", the field will be not indexed by the Indexing Broker, and will not be accessible to TSWeb users when they are searching.)
faceton_yn	nchar(1)	"Y" in this field will mark the field to be facetable in Solr searches. TSWeb users will be able to refine their search results by selecting values from the metadata field. When set to "N", faceting will not be available for the field. Max_length of these fields should be less than 250 characters, and preferably controlled vocabulary fields (through the POPUPs table) to limit the number of facet entries.
prompt_yn	nchar(1)	This field is intended for future use.
lang_id	nchar(10)	Identifies the language (from the language_local table, lang_id column). For example, en_US.
parent_id	smallint	This field is intended for future use.

Field	Data Type	Description
bucket_type	nvarchar(30)	For Refine Search (faceting), defines the type of bucketing, defined by the data type (timestamp, integer).
bucket_size	real number	For Refine Search bucketing, defines the size of each facet/range.
min_start	datetime	For Refine Search bucketing involving timestamps, shows the start value shown to users.
max_end	datetime	For Refine Search bucketing involving timestamps, shows the end value shown to users.
min_start_int	real number	For Refine Search bucketing involving integers, shows the minimum value shown to users.
max_end_int	real number	For Refine Search bucketing involving integers, shows the maximum value shown to users.
time_range	nvarchar(32)	This field is intended for future use.
translatable_yn	char(1)	"Y" in this field will mark the field as translatable, for multi-language implementation. When set to "Y", Telescope will look for translated values in the COLUMN_DISPLAY table.

EXTRA_POPUPS (Deprecated)

DEPRECATED.

ICONIC_FIELDS

The ICONIC_FIELDS table contains information about the iconic field data (metadata fields) in the Telescope database. The following chart describes the columns in the ICONIC_FIELDS table.

Field	Data Type	Description
id	integer (identity)	Unique ID generated automatically on insert into the table. In Oracle, this is a plain integer column with an insert trigger to populate it from an Oracle sequence.
column_idx	integer	Cross-reference to theID field in the *EXTRA_COLUMNS* table for the column the popup menu is associated with.
position	integer	The position in the sequence (1, 2, 3 and so on)
value	Integer	The value shown to the user.

icon	binary	The icon image. This is stored in PNG format (to allow for transparency), of any size.

POPUPS

In the Telescope environment, you can configure a metadata field to use an assigned popup menu of values (or list of values) to enforce a controlled vocabulary during data entry. Use a popup menu to restrict the values a user can enter in the field or simply to assist them in entering data. This activity can be done using the TSAdmin interface (refer to the section on popup menus in the Administrator's Reference Manual).

A limited amount of formatting data can be embedded in the values for display purposes within the clients, as follows:

Separators: If the first character of the text is a '-', then the rest of the text in the item will be ignored, and the item treated as an un-selectable separator item in the popup menu (usually represented as a grey horizontal line or dotted line).

Text Style: The character '<', if it appears in the text, will be followed by one of the following: 'B', 'I', or 'U', representing bold, italic, or underline, respectively. If the '<' is followed by any other character, it will be removed from the text before insertion into the popup menu (this effectively precludes the use of the '<' character in the text).

The following chart describes the columns in the POPUPS table.

Field	Data Type	Description
column_idx	short integer	Cross-reference to the ID field in the *EXTRA_COLUMNS* table for the column the popup menu is associated with.
popup_idx	short integer	Index representing the text's position in the popup menu. Popup indices are numbered from 1 and are contiguous through to the maximum number of items on the menu.
cascade_column	short integer	If not zero, this value is a cross-reference to the ID field in the *EXTRA_COLUMNS* table for the column this popup cascades from.
cascade_value	short integer	If not zero, the value of popup_idx for the column defined by cascade_column that must be selected in order for this item to be active.
popup_text	nvarchar (255)	Value to be used in the popup list. See above for information on embedding style or separator meta-tags in this text.

Popup menu values can 'cascade' off popup menu items in other fields. For example, assume a metadata model with 'Country' (id=1 in extra_columns) and 'State' (id=2 in extra_columns) fields. Country has a popup menu that contains the values 'USA' and 'Canada'. This would be represented in the popups table as:

column_idx	popup_idx	cascade_column	cascade_value	popup_text
1	1	0	0	USA
1	2	0	0	Canada

The metadata model requires that the popup of the State field be set up such that, if 'USA' is selected for Country, the State field's popup menu should have the values 'New York' and 'California' in it; and if 'Canada' is selected for Country, the State field's popup menu should have the values 'Ontario' and 'British Columbia' in it. To achieve this, cascading popup menu values are defined as follows:

column_idx	popup_idx	cascade_column	cascade_value	popup_text
2	1	1	1	New York
2	2	1	1	California
2	3	1	2	Ontario
2	4	1	2	British Columbia

POPUPS_LANG

Default language pop-up menu values are stored in the POPUP table of the Telescope database. When users are viewing the interface in another language, popup menu choices in the user's selected language are retrieved at run time from the POPUPS_LANG table and displayed in the TSWeb interface. Popup menu values remain in the same order, regardless of which language they are being viewed in.

Note: A SOAP API call is required to populate this table. See the *Administrator's Guide*.

The following chart describes the columns in the POPUPS_LANG table.

Field	Data Type	Description
column_idx	small integer	The ID value of the popup menu being defined, as identified by the column_idx column of the POPUPS table.
popup_idx	small integer	Index representing the text's position in the popup menu. Popup indices are numbered from 1 and are contiguous through to the maximum number of items on the menu.

Field	Data Type	Description
lang_id	nvarchar(10)	The language ID of the values in the popup_text array. (For example, "fr_CA".) This value must exist in the lang_id column of the language_local table.
popup_text	nvarchar(256)	Contains a comma-separated sequence of strings that are the translations of strings defined in the popup_text column of the POPUPS table. All strings must be included that have the same ID value in the *POPUPS* table (as identified by the column_idx of the POPUPS_LANG table), and must be ordered to match the popup_idx values in the POPUPS table.

Asset Data Tables

All of the tables listed below are described in detail in the sections that follow.

The following tables manage Telescope asset information:

- ACCESS_HISTORY (tracks user actions)
- CHECKOUTS (contains information about assets checked out of the database)
- DOC_LINKAGES (stores the contents of container fields)
- DOC_RENDITIONS (stores file attribute information about assets)
- EDITORIAL (contains metadata assets)
- ED_VERSIONS (stores non-primary versions of an asset)
- FT_CONTENTS (stores full-text data for assets used for full-text search and retrieval functions)
- INOTES (stores notes users add to assets) manages the Telescope asset version control system).
- THUMBNAILS (contains binary encrypted data that portrays the thumbnail version of each asset)
- VIEWEX (contains binary data–raster graphics images–used to display the extended view of an asset)

The following tables deal with COV extended views:

- COV_FONTS
- COV_GEOMETRY
- COV_INFO
- COV_MAJORTYPES
- COV_PAGES
- COV_SECTIONS
- COV_SECTIONTYPES

The following tables handle Video Manager views:

- VL_ANNOTATIONS
- VL_ANNOTATIONSETS
- VL_CLIPS
- VL_INFO

- VL_PLAYLISTS
- VL_PROXIES
- VL_TEXT
- VL_THUMBNAILS
- VL_TRACKS

The following tables contain metadata that handles extended views:

- EMBEDDED_METADATA
- IANNOTATION
- ZOOM_INFO

ACCESS_HISTORY

The ACCESS_HISTORY table tracks user actions in the Telescope environment. It records every action by individual users against each asset.

This table is one of the most useful Telescope tables when creating application customizations. Database triggers can be created on this table to perform database tasks based on the action a user performed. For example, a trigger could be created to send an email to the Telescope administrator every time a user deletes an asset from the system.

In very active Telescope environments, the ACCESS_HISTORY table can grow very quickly. To prevent database fragmentation and to distribute I/O, consider physically storing the ACCESS_HISTORY table separately. (In an Oracle environment, use a separate tablespace. You may want to consider placing the ACCESS_HISTORY and TS_STATISTICS tables together in one tablespace.)

The following chart describes the columns in the ACCESS_HISTORY table.

Field	Data Type	Description
id	identity	Unique ID generated automatically on insert into the table. In Oracle, this is a plain integer column with an insert trigger to populate it from an Oracle sequence.
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
access_time	datetime (timestamp)	Date and time the access was performed. A NULL in this field indicates that the action is an approval, which has not happened yet.

Field	Data Type	Description
access_user	nvarchar(32)	User name of the user who made the access. A NULL in this field indicates that the action is an approval any user can take advantage of, which has not happened yet.
access_approval	nvarchar(255)	Name of the user who approved the access. If the access_type entry is 99, this field contains a text description of the access type being defined.
access_type	small integer	A number representing the type of access performed by a user or process when manipulating an asset record or associated files. Possible values include: **1** Download **2** Editorial Modification **3** Delete **4** Insert (Archive) **5** Synchronize **6** Move File(s) **7** Open in Application (Not used in Version 9.2 or later) **8** Check Out **9** Check In **10** Attach Rendition **11** Order Asset **12** Extended View **13** Native Plugin **14** Seen in Hot Folder **16** Entered Hot Folder Queue **17** Menu **21** Job ingested successfully by Telescope Uploader **22** Job ingested with warning (incomplete) by Telescope Uploader **99** Custom
access_description	nvarchar(255)	Free-text descriptive field.
rendition	integer	Rendition ID of the document's file that was affected by this activity, if applicable. Zero or NULL otherwise.

CHECKOUTS

The CHECKOUTS table contains information about documents (assets) that are checked out of the database. Records are added to this table when a user checks out a file and removed from the table when the file is

checked back in. While an asset is checked out, the physical file cannot be checked out by any other Telescope user. This is how version control and workflow are enforced through the Telescope application.

You can create customizations to automate workflow and the checkout processes. For example, imagine an environment where the metadata of an asset changes as it progresses through its lifecycle. User A is the current owner of the asset and changes the "status" field of the asset from "in-progress" to "review" and he changes the "owner" to user "B". A trigger may fire that automatically checks the asset out on behalf of user B to lock the asset, preventing any further changes by other users. When user B is finished, she checks the asset back in and changes the "status" and "owner" fields accordingly.

The following chart describes the columns in the CHECKOUTS table.

Field	Data Type	Description
record_id	integer	Cross-reference to the record_id in the EDITORIAL table of the asset that is checked out.
user_name	nvarchar(32)	Name of the user (taken from the USERS table) who performed the checkout.
file_name	nchar(255)	Name of the file that has been checked out, taken from the local copy of the file being worked on.
chkoutdate	datetime (timestamp)	Date and time when the checkout was performed (this will come from the {fn NOW()} scalar function on the database, so that it will be a server-based time).
chkoutfile	nvarchar(2000)	file_location for the checked-out file on the user's machine. The format of this field is identical to the file_location field in the EDITORIAL table. This field refers to the local copy of the file that is being worked on, not the original, which is referred to by file_location in the EDITORIAL table.
filemoddate	datetime (timestamp)	Modification date and time of the file. It is extracted from the local copy of the file after it is downloaded to the user's machine. This date and time are compared to the file's modification date and time when the file is checked back in to the database.
rendition	integer	Rendition ID of the document's file that was checked out. This will be used during check-in to ensure that the proper rendition is versioned.

COV_FONTS

The COV_FONTS table stores information about the fonts used in a COV document. The following chart describes the columns in the COV_FONTS table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
font_index	integer	Place of the font in the list for the COV. This value ranges from 1 to n, where n is the number of fonts recorded for a COV.
font_name	nvarchar(255)	Human-readable name of the font.
font_code	nvarchar(255)	Machine name of the font, if applicable. Some fonts have a display name and a name by which the font is recognized by the OS. For example, "Helvetica" is often called "HELVE" in the operating system.

COV_GEOMETRY

The COV_GEOMETRY table represents items of interest on a given page of a COV. In the COV structure, a COV display is divided up into pages, with each page having a preview and some number of "Geometry" items on the page. Each geometry is either a "document" geometry that refers to a sub-document in the COV (and therefore has another asset record for it in the database), or a "text" geometry that refers to a text area on the page (and therefore has an FT_CONTENTS record for it in the database). The following chart describes the columns in the COV_GEOMETRY table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
page_num	integer	The number of the page within the component object. These pages are numbered sequentially from 1 to n, where n is the number of pages contained in the num_pages field in the cov_info table for this COV.
geometry_order	integer	Order of the geometry on the page. Geometry items are ordered from 1 to n, where n is the number of geometry items on the page. This ordering becomes especially important when there are overlapping geometry items on a page, in which case the ordering is used to provide "front-to-back" layering of the geometry items.

Field	Data Type	Description
geometry_type	nchar(10)	Text value that indicates the type of the geometry entry. There are two valid geometry types: **document** - refers to a sub-document in the COV (for example, a placed art file in a Quark document). There will be another DOC_RENDITIONS entry in the database for this placed file. **text** - describes an area of text on the page. The contents of the text area will be described in the FT_CONTENTS record that is associated with this geometry.
enclosing_left	integer	Left side of the rectangle that encloses this geometry on the page. **Note:** The enclosing rectangle for a geometry assumes a normalized page dimension of (0, 0, 1000, 1000), regardless of the physical size of the page. This means that all of the enclosing rectangle columns will contain values in the range of 0 to 1000.
enclosing_top	integer	Top side of the rectangle that encloses this geometry on the page. See note above.
enclosing_right	integer	Right side of the rectangle that encloses this geometry on the page. See note above.
enclosing_bottom	integer	Bottom side of the rectangle that encloses this geometry on the page. See note above.
file_path	nvarchar(max)	Full file path of the sub-document's file. This can be in any representation and is usually represented in the format native to the OS on which the parent document was created.
create_date	datetime (timestamp)	Date the sub-document's file was created. Note: The original system create and modify dates may not always be preserved when a file is imported, because of differences in data handling between different operating systems.
mod_date	datetime (timestamp)	Date the sub-document's file was last modified. Note: The original system create and modify dates may not always be preserved when a file is imported, because of differences in data handling between different operating systems.
file_size	big integer	Size (in bytes) of the sub-document's file.
fulltext_id	integer	Cross-reference to the ft_id column in the FT_CONTENTS table for a 'text' geometry. The text of the geometry item will be in the FT_CONTENTS table with the given ID.

Field	Data Type	Description
file_checksum	nchar(32)	A hexadecimal representation ("hash") of the file. This "unique" hash is created from the first 2 MB and last 2 MB of the file, a variation of the MD5 function. This representation is sometimes referred to as an "MD5" in Telescope functions, database model names, and so on but is called "checksum" in this document. This information is used to link placed art back to the parent document. If the file cannot be found, this entry will be NULL.
xmp_docid	nvarchar(36)	If Adobe InDesign files are imported using an older version of the Xinet plugin, the checksum information may not be populated into the file_checksum field. Instead, similar document identification information is stored in this xmp_docid field.

COV_INFO

The COV_INFO table contains information about the component object view (COV) displays. If a document in the Telescope database has a COV display for its extended view, that document's entry in the VIEWEX table will contain NULL for the viewex field, and the text "COVv" in the data_type field. For such a document, there will be a single COV_INFO entry, which contains general information about the COV display and acts as the "root" of the tree of entries in the COV that describe the pages, text, geometry items, etc. that make up the COV display. The following chart describes the columns in the COV_INFO table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.

Field	Data Type	Description
dflt_display	nchar(10)	Default display layout for this COV. If this field has a value, it overrides the user's preference for display type for this particular COV. The possible values for this field are: **<empty>** - the user's display preference should be used to view the COV **1UP** – the COV is displayed 1-up by default. **2UP_ODD** – the COV is displayed by default as 2-up, with odd pages on the right. **2UP_EVEN** – the COV is displayed by default as 2-up, with even pages on the right. **THUMBS** –used for PPT/PPTX COV previews. If the DATA_TYPE is COVv and this field is THUMBS, it indicates there is a Powerpoint (PPT or PPTX) to preview. The pages stored in COV_PAGES then have a slightly different format in the database: odd pages are created for the thumbnails, and even pages for the preview slides. (If the DATA_TYPE is COVv and this field is not THUMBS, the preview is simply displayed as pages of a document.)
cov_description	nvarchar(255)	Optional description of the COV, which is shown in the COV display in Telescope. This description could be used, for example, to show the name and version of the application that created the document the COV belongs to.

COV_MAJORTYPES

The COV_MAJORTYPES table stores the common types of top-level sections in a print publication. These sections are used to create table of contents (TOC) entries for COV documents. The following chart describes the columns in the COV_MAJORTYPES table.

Field	Data Type	Description
type_id	integer	A number representing the type of section the TOC entry represents.
type_name	nvarchar(256)	A description of the type of section the TOC entry represents.

COV_PAGES

The COV_PAGES table contains the graphical representations of each page in the component object view (COV). The following chart describes the columns in the COV_PAGES table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
page_num	integer	Number of the page in the component object. These pages are numbered sequentially from 1 to n, where n is the number of pages contained in the COV.
page_height	integer	Height of the physical page in the file represented by the COV. This is an integer representing the dimension in points (1 inch = 72 points).
page_width	integer	Width of the physical page in the file represented by the COV. This is an integer representing the dimension in points (1 inch = 72 points).
pvw_pix_height	integer	Height of the page preview, in pixels, for this page of the COV. In other words, when decompressed, the graphic data in the PREVIEW field will be this many pixels high.
pvw_pix_width	integer	Width of the page preview, in pixels, for this page of the COV. In other words, when decompressed, the graphic data in the PREVIEW field will be this many pixels wide.
preview_type	nchar(4)	Four-character type indicating the format of the page preview. At present, it will contain "JPEG".
preview	binary	Preview of the page. This field is a "blob" field whose data is interpreted by the value in the preview_type field.

COV_SECTIONS

The COV_SECTIONS table represents the table of contents entries in COV documents. The following chart describes the columns in the COV_SECTIONS table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
page_num	integer	The sequential number of the page in the COV document. These pages are numbered sequentially from 1 to n, where n is the number of pages contained in the num_pages field in the cov_info table for this COV. In BISG terms the sequence page number would correspond to an Absolute-PageNumber.
section_index	integer	A number from 1 to n where n is the number of section items on the page. This number indicates the order of section items on the page in case there is more than one section defined per page.

Field	Data Type	Description
section_pos	integer	The coordinate relative to the top margin of the page on the Y axis in points.
section	nvarchar(256)	The section text.
section_type	integer	The type_id from the COV_SECTIONTYPES table corresponding to the type of section the TOC entry points to.
indent_level	integer	The heading level of the TOC entry in the table of contents.
page_numreal	nvarchar(256)	The page number as it appears on the page in the COV document. This may be different from the page_num value if front matter pages are un-numbered or use roman numerals.
major_type	integer	The type_id from the COV_MAJORTYPES table corresponding to the type of major section this TOC entry points to.

COV_SECTIONTYPES

The COV_SECTIONTYPES table stores the common types of sections in a print publication. These sections are used to create table of contents (TOC) entries for COV documents. The following chart describes the columns in the COV_SECTIONTYPES table.

Field	Data Type	Description
type_id	integer	A number representing the type of section the TOC entry represents.
type_name	nvarchar(256)	A description of the type of section the TOC entry represents.

DOC_FILE_INFO

This internal table stores metadata extracted from files by the File Info I-Piece. This metadata is configurable, but will vary by the type of file that was imported (for example, metadata from a video file will be different from that from an audio stream or audio file, and from that for an image file).

The following chart describes the columns in the DOC_FILE_INFO table.

Field	Data Type	Description
id	integer	Unique ID generated automatically on insert into the table. On Oracle, this is a plain integer column with an insert trigger to populate it from an Oracle sequence.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
rend_id	integer	Cross-reference to the RENDITIONS table, which indicates which rendition this record applies to.
keystr	nvarchar(64)	Usage varies by file type. For image files: width. For video files: codec.
valuestr	nvarchar(255)	Usage varies by file type. For image files: 640. For video files: avc.
units	nvarchar(24)	Usage varies by file type. For image files: pixels. For video files: NULL.

DOC_LINKAGES

The DOC_LINKAGES table supports the Telescope container data type field, which is a type of field that contains other assets that may be linked to the selected (or containing) asset.

- A container field is indicated in Telescope in the EXTRA_COLUMNS table when data_type = 12.
- When a container field is created, its column in the EDITORIAL table is created as an integer field. Rather than holding the actual contents of the field, as with other data types, this column holds a counter for the number of assets linked in the container. This counter changes to reflect the accurate count of contained assets as they are added or removed from the container field.
- The DOC_LINKAGES table contains one row for each asset in the container field.

Container fields can be created for users to populate freeform by drag and drop, or they can be created to be populated automatically with linked art when multi-page (Component Object View, or COV) documents are imported (for example, by the InDesign I-Piece). COV links are maintained if the *Maintain COV Links* setting is checked in TSAdmin; for details, see the *Telescope Administrator's Reference Manual*.

Example:

If there are 32 assets in a container field called "placed_art":

- The "placed_art" row in EXTRA_COLUMNS data_type = 12.
- The "placed_art" column in EDITORIAL is "32"
- There are 32 rows in the DOC_LINKAGES table, each specifying the record_id for each of the placed_art parent (containing) asset, its ID value in the EXTRA_COLUMNS table, and the record_id for one of its

child (contained) assets. (The table also a column to specify the order the child asset will appear in the list.)

Notes:

- When creating customizations that deal with container fields, be careful to increment and decrement the container count correctly in the EDITORIAL table. Consider placing an on-insert and delete trigger in the DOC_LINKAGES table to increment the container count in the EDITORIAL table automatically.
- The DOC_LINKAGES table can become quite large (millions of entries) very quickly, so performing a select count or a max function against the table to adjust the container count is not advised.

The following chart describes the columns in the DOC_LINKAGES table.

Field	Data Type	Description
parent_id	integer	RECORD_ID of the containing or parent record (asset) in the EDITORIAL table.
column_id	integer	ID value of the container field as listed in the EXTRA_COLUMNS table. This value indicates the container field this entry (parent or child relationship) applies to.
link_order	integer	Order of the thumbnail representations of each asset in the container field when displayed in the Document Info view.
child_id	integer	RECORD_ID of the contained or child record (asset) in the editorial table. A thumbnail representation of this asset appears in the container field of the parent asset when viewed from the Document Info view. Note that the child_id column may contain multiple references to the same asset, since multiple container fields can each contain the same child assets.

DOC_RENDITIONS

The DOC_RENDITIONS table stores file attribute information about a particular asset. This information is populated by Telescope automatically during ingestion. The record_id field ties the entries in this table back to those in the EDITORIAL table. The relationship between EDITORIAL and DOC_RENDITIONS is one to many (or one to none). If the entry in EDITORIAL represents a metadata placeholder ("New Document" for example) without any physical files, there will not be an entry in DOC_RENDITIONS. For example, there may be a Telescope asset of type "PROJECT" that contains metadata information about a particular project but does not link directly to a physical file. Conversely, there may be a Telescope asset that is from a photo shoot. The EDITORIAL record could contain the metadata information about the shoot, the photographer, and a description

of the picture. The DOC_RENDITIONS table could have three entries: one for the low-resolution rendition of the picture, one for the medium resolution rendition, and one for the high-resolution rendition of the picture.

The following chart describes the columns in the DOC_RENDITIONS table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
rend_id	integer	Cross-reference to the RENDITIONS table, which indicates which rendition this record applies to.
file_location	nvarchar(2000)	(Legacy field) Physical file location for the rendition. **Note:** It is still possible store the physical location of the file outside the Telescope installation, but this legacy feature is discouraged. We encourage Telescope installations to import data through the Telescope Uploader so that the data is stored properly and proper checks are performed.
long_name	nvarchar(2000)	Viewable name (path) for the file that is used to populate the popup menu in the Document Info window for file path and for user searches on file path.
file_type	nchar(4)	Type of file that is referred to by this rendition (e.g., "TIFF" or "ESPF"). For display purposes, this field is used to map into the TYPE_CODES table, which provides a "full-text" description of the file types.
file_name	nchar(255)	Simple name of the file without any preceding path.
file_size	big integer	The size, in bytes, of the original file, which can be used to determine download times. This field is a 'bigint' database type, which means that Telescope can accurately represent files whose size is up to 264 bytes (8 million terabytes). Note: These file sizes may differ from the actual file sizes.
file_info	nvarchar(255)	Free-form text filled-in during ingest by the I-Piece that reads the file or by Telescope if the file is a graphic. It contains general information about the file, such as the resolution and color depth for images or the sample size and sample rate for audio assets. This field is displayed as-is to the user in the Editorial View. The format of this information is not standardized and can change from one Telescope version to the next.
create_date	timestamp	Date the physical file was created on the disk. It is not user-editable and contains the actual created date of the physical file. Note: The original system create and modify dates may not always be preserved when a file is imported, because of differences in data handling between different operating systems.

Field	Data Type	Description
mod_date	timestamp	Date the physical file was last modified (as known when the file was put into the database). There is the danger that this field will get out of date if the user modifies the document outside of Telescope, but it can be resynchronized through the use of the "Synchronize Documents" option. Note: The original system create and modify dates may not always be preserved when a file is imported, because of differences in data handling between different operating systems.
file_checksum	nchar(32)	The hexadecimal representation of the checksum for the file. If the file cannot be found, the entry will be NULL. Checksum information is used to link placed art back to the parent document. Note: This checksum information is customized by Telescope and cannot be compared with MD5 generated by other tools.
lang_id	nchar(10)	Identifies the language (from the language_local table, lang_id column). For example, en_US.
xmp_docid	nvarchar(36)	If Adobe InDesign files are imported using an older version of the Xinet plugin, the checksum information may not be populated into the file_checksum field. Instead, similar document identification information is stored in this xmp_docid field.

ED_VERSIONS

The ED_VERSIONS table stores non-primary versions of an asset. When an asset is checked in to Telescope, the existing asset information is moved to the ED_VERSIONS table, to indicate that it has been versioned. The new asset's information is stored in the DOC_RENDITIONS table, as it is now the current version of the asset. When a new version is created it retains the same filename as the original filename. But the new entry in the ED_VERSIONS table will have "-x" appended to its filename (where x is the version number). When you download an asset, you will always get the most recent version of the asset. Note that, when a new version is added to the system, the thumbnail of the asset is updated to be the same as the latest version added.

Users with special permissions can see, download, and promote a version of an asset. The promoted version will be the primary version and its version will be updated to reflect the new (updated) version. The following chart describes the columns in the ED_VERSIONS table.

Field	Data Type	Description
version_id	integer	Unique ID generated automatically on insert into the table. In Oracle, this is a plain integer column with an insert trigger to populate it from an Oracle sequence.
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table for that file.
rend_id	integer	Cross-reference to the RENDITIONS table, which indicates which rendition this record applies to.
file_location	nvarchar(2000)	Physical file location for the rendition.
long_name	nvarchar(2000)	Viewable name (path) for the file that is used to populate the popup menu in the Document Info window for file path and for user searches on file path.
file_type	nchar(4)	Type of file that is referred to by this rendition (e.g., "TIFF" or "ESPF"). For display purposes, this field is used to map into the TYPE_CODES table, which provides a "full-text" description of the file types.
file_name	nchar(64)	Simple name of the file without any preceding path.
file_size	big integer	The size, in bytes, of the original file, which can be used to determine download times.
file_info	nvarchar(255)	Free-form text field filled in during ingest by the I-Piece that reads the file or by Telescope if the file is a graphic. It contains general information about the file, such as the resolution and color depth for images or the sample size and sample rate for audio assets. This field is displayed as-is to the user in the Editorial View. The format of this information is not standardized and can change from one Telescope version to the next.
create_date	timestamp	Date the physical file was created on the disk. It is not user-editable and contains the actual created date of the physical file. Note: The original system create and modify dates may not always be preserved when a file is imported, because of differences in data handling between different operating systems.

Field	Data Type	Description
mod_date	timestamp	Date the physical file was last modified (as known when the file was put into the database). This field will get out of date if the user modifies the document outside of Telescope, but it can be resynchronized using the "Synchronize Documents" functionality. Note: The original system create and modify dates may not always be preserved when a file is imported, because of differences in data handling between different operating systems.
file_checksum	nchar(32)	The hexadecimal representation of the Checksum for the file. If the file cannot be found, the entry will be NULL. Checksum information is used to link placed art back to the parent document.
vcversion	nchar(16)	The version number of the asset, used during checkout when several versions of the same file are being stored. This version number corresponds to the number added as a suffix to the file name. (For example, the file Filename-2.docx has vcversion "2".)
chkindate	datetime (timestamp)	The timestamp of when the version was checked in.
thumbnail	binary	The thumbnail image of the version, copied from the THUMBNAILS table (not the Graphics Broker).
xmp_docid	nvarchar(36)	If Adobe InDesign files are imported using an older version of the Xinet plugin, the checksum information may not be populated into the file_checksum field. Instead, similar document identification information is stored in this xmp_docid field.

EDITORIAL

The EDITORIAL table is one of the most important tables in the Telescope database. This table contains a set of default system-wide metadata for each asset, including the record_id for an asset, which is the link for all the other metadata tables. It will also usually contain a large part of the user-defined metadata model. .

To appear as metadata fields in the Telescope environment, each new column must be included as a row in the EXTRA_COLUMNS table. The easiest way to ensure this is all set up correctly is to add new metadata fields through the Fields tab in the TSAdmin interface. This will ensure all tables are updated correctly.

Data can also be stored in tables other than the EDITORIAL table; moreover, external tables are required for normalized repeating fields. For details on how to set up external tables, see the instructions included in the section for the EXTRA_COLUMNS table.

Note: Telescope Administrator is not aware of any database sizing limitations on tables. In a SQL Server database, for example, the width of a database table (the sum of the data contained in one row) cannot exceed 8060 bytes (not for tables that contain varchar, nvarchar, varbinary, or sql_variant though). In environments where the sum of the metadata to be added to the Telescope environment exceeds the storage limitations of the EDITORIAL table, the data must be normalized into separate tables.

Telescope queries that span external metadata tables take the form of an outer joined select statement. In environments where the underlying database does not support the outer join functionality, these queries are achieved through a series of nested selects. Telescope provides users with an ad hoc query tool that can process a wide range of statements against the database. In most environments, there is a general use trend by the users. You should monitor the system and tune the underlying Telescope tables to match users' behavior. Tuning may include, but is not limited to, adding indexes, functional indexes, rebuilding indexes, distributing data storage across physical drives, de-fragmenting data storage, and adjusting database system environment settings.

The following chart describes the default columns in the EDITORIAL table before it is customized. These columns are on the editorial table by default and are specifically used for functionality within Telescope. There will be many more columns in the table after a customer's metadata model is defined.

Field	Data Type	Description
record_id	integer	Unique identifying code for the record, which is used to join the table to subsidiary tables.
member_yn	nchar(1)	*DEPRECATED.*
group_yn	nchar(1)	*DEPRECATED.*
status	nchar(1)	A value indicating the checkout status of the document, with the following values: **NULL** – means the file is checked in and available for checkout. **Y** – means the file is checked out for modification and cannot be checked out by anyone else.
vcversion	nchar(16)	*DEPRECATED.*
chkindate	datetime (timestamp)	*DEPRECATED.*
vcparent	integer	Used internally by Telescope to link 'derivative' assets (i.e. those created by 'Promoting' a version or a video clip to a sub-asset) to their parent assets.

Field	Data Type	Description
versioned	nchar(1)	*DEPRECATED.*
approvpend	nchar(1)	Determines whether or not the document is pending approval (and therefore hidden). If the flag is "Y", the document is hidden from users. If the flag is NULL (its normal state), the document is visible.
(Customer defined fields)	(Customer defined)	Any field created within TSAdmin used by the customer.

EDITORIAL SELECTION

The EDITORIAL_SELECTION table is used by the Telescope system to keep a temporary list of assets currently selected by the user. **Do not alter this table in any way.**
The following chart describes the columns in the EDITORIAL_SELECTION table.

Field	Data Type	Description
instance_id	nvarchar(10)	The ID for the instance running the application (in a clustered environment).
user_session_id	nvarchar(32)	A unique session ID assigned to the client's session when the user logs in (from the Session Broker).
user_component_id	integer	The incremented ID for the UI component.
record_index	integer	The order the record will appear in a container field.
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.

EMBEDDED_METADATA

The embedded_metadata table stores embedded metadata (XMP,IPTC,etc) that was extracted by the Metadata I-Piece. The table is essentially a key-value pair, storing a tag for the extracted metadata and the value for that tag.

The following chart describes the columns in the EMBEDDED_METADATA table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.

Field	Data Type	Description
tag	nvarchar(256)	The embedded tag. For example, • File property tags (the file's size, for instance) • XMP tags (the document ID, for instance) • PDF tags (the PDF version, for instance)
value	nvarchar(2000)	The values of the embedded metadata, as extracted for the corresponding tag.

FT_CONTENTS

The FT_CONTENTS table provides full text searches on content in the Telescope environment. Telescope I-Pieces for asset file types that contain text are responsible for parsing the text content out of the document (Word, Quark, etc.) and populating the FT_CONTENTS table. There is a context index on the FT_CONTENTS table, specifically on the FT_TEXT column. The syntax of full text queries submitted from Telescope are formatted as follows (Oracle example):

```
select ft.record_id, ft.ft_text, score(1) from dbo.ft_contents ft, dbo.editorial
ed where ft.record_id = ed.record_id and contains(ft.ft_text, 'about(author)',
1) > 0 order by score(1) desc
```

The select statement above returns all entries containing the word "author". The following chart describes the columns in the FT_CONTENTS table.

Field	Data Type	Description
ft_id	integer	Unique identifier for the content entry.
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
ft_changestamp	timestamp	The date and time the record was changed.
ft_lang	nchar(10)	This field is intended for future use.
ft_text	nvarchar(max)	Plain text contents of the entry.

IANNOTATION

The IANNOTATION table stores information about annotations added to an asset. The following chart describes the columns in the IANNOTATION table.

Note: Do not edit this autopopulated table; your changes will be overwritten.

Field	Data Type	Description
user_name	nvarchar(32)	The user who added the annotation.
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
mark_up	nvarchar(max)	The text of the annotation.
last_change	timestamp	The date and time the annotation was created and/or updated.
page_num	integer	The page number within the assets where the annotation is added.
lang_id	nchar(10)	Identifies the language (from the language_local table, lang_id column). For example, en_US.
marker	integer	An integer field used to mark the location of the annotations. This enables users to request individual page images and geometry information for any page of a COV preview. The units are based on the asset type. For example: For COV annotations, represents the page number For video annotations, represents the millisecond offset from the beginning of the digital video.

INOTES

The INOTES table stores the notes users add to assets in Telescope. The following chart describes the columns in the INOTES table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.
note_date	datetime (timestamp)	The date-time stamp when the note was added.
note_user	nvarchar(32)	The user who added the note.
note_text	nvarchar(max)	The note text.
lang_id	nchar(10)	Identifies the language (from the language_local table, lang_id column). For example, en_US.

THUMBNAILS

Thumbnails appear when the user browses through the collection (catalog) view of the assets in Telescope. Every entry in the EDITORIAL table should also have an entry in the THUMBNAILS table. This table contains the binary data that portrays the thumbnail version of the asset. Each asset has a single thumbnail representation, regardless of how many renditions of the asset exist in the system. A NULL entry for the thumbnail data is valid. The binary data is stored as a 128 x 128 pixel 72 DPI JPEG stream that is by default encrypted.

When a new file is added to Telescope its thumbnail is added to the THUMBNAILS table by the Graphics Broker. The Telescope I-Pieces are responsible for the supported formats. You can add different I-Pieces to the Graphics Broker to enable the support for different file formats. If you import a file type that is not supported by Telescope, the thumbnail image will be set to a default image for that file type, as defined in the TSAdmin application and stored in the type-codes table. If no such entry exists, the thumbnail will default to an overall thumbnail (also defined in Telescope). Encrypted by default but can be stored encrypted or unencrypted by toggling the ENCRYPT_PREVIEW setting in the Graphics Broker registry keys. Changing that setting only changes the future ingests.

The following chart describes the columns in the THUMBNAILS table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.
file_name	nchar(255)	Simple name of the file without any preceding path.
thumbnail	long raw (varbinary)	Encrypted binary data for the thumbnail that is displayed in Telescope. Note: It is possible to put your own 128 x 128 pixel, 72 DPI JPEG thumbnail into the thumbnail table for any asset. However, if the asset is synced this thumbnail will be replaced with the one defined by the poplulating I-Piece.

VIEWEX

The VIEWEX table specifies the data type for each asset, and this determines the type of extended view (preview) that will be shown to TSWeb users for that asset.

For image files (data_type "JPEG"), this table stores a medium-sized binary representation in the table's viewex field. For all other types of files, the preview data is stored elsewhere For example, in the DOC_RENDITIONS table for video renditions, or in the COV_PAGES table for component object view (COV) files, which are files that have multiples pages associated with them (like Office, Quark, and InDesign documents).

The exact specifications for the representation for each data type is determined by the XML ("prefsML") file of the I-Piece that is interpreting and populating the data. For details on XML file settings, see the prefsML section in the manual for the respective I-Piece.

The following chart describes the columns in the VIEWEX table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.
file_name	nchar(255)	*DEPRECATED.*

Field	Data Type	Description
data_type	nchar(4)	Defines the file type, which is used to determine the type of preview shown to the user on TSWeb. This information is provided by an I-Piece when it populates the viewex table. Possible values include the following: **COVv**: The asset requires a multi-page component Object View (COV) preview. The format of these previews is defined in the COV_INFO table, and the data for these previews is stored in the COV_PAGES table. **JPEG**: The asset is an image file, and an image preview will be shown. The image preview is stored as a smaller JPEG representation in the viewex field of this VIEWEX table. **MP3 or MP4**: The asset preview is an audio (MP3) or a video (MP4) preview. Other audio/video file types are: **ViRa**: A Video Manager (VM) preview. **FLSH**: An Adobe Flash preview. **TPlt**: A preview created from a Telescope Playlist. Unless they are explicitly identified as FLSH, TPlt, or ViRa, all audio/video files are given data types MP3 (for audio) or MP4 (for video with or without audio). For MP3/MP4 data types, previews play back in the HTML5 player without the additional VM functions of keyframes, clips, and so on. Audio/video previews are stored in the DOC_RENDITIONS table. Their proxy is identified by the rendition ID associated with the ip_viravideorendition (for VM) or ip_proxyrendition (for non-VM) keywords in the DB_SETTINGS table (see the descriptions in that table for details). This rendition ID can be either a valid rend_id cross-reference to the RENDITIONS table, or a URL if ip_viravideorendition or ip_proxyrendition is configured with 'vl_proxies.url.'

Field	Data Type	Description
viewex	binary	When the data_type field is "JPEG", this field contains a binary JPEG version of the image, , which is typically a medium resolution JPEG file of 512 x 512 pixel resolution, 72 DPI, but can be configured to 1024x1024. If the data_type field is not "JPEG", this field has a NULL value. The I-Piece that handles the file is responsible for determining and interpreting the contents of this field.. Note that this extended view data can be encrypted, as determined by the Graphics Broker registry key ENCRYPT_PREVIEW.

VL_ANNOTATIONS

The VL_ ANNOTATIONS table contains the annotation buttons created by the user. The following chart describes the columns in the VL_ ANNOTATIONS table.

Field	Data Type	Description
button_id	integer	A unique identifier for the button.
set_id	integer	The ID of the annotation set the button belongs to.
texttrack_id	integer	The ID of the text track the annotation will be written to.
button_type	integer	Not currently in use.
mark_type	integer	Indicates whether the button is a mark in or a mark out button.
name	nvarchar(256)	The name of the button as it appears in the Telescope interface.
description	nvarchar(256)	The annotation text that will be written to the text track.
staticpopupdata	nvarchar(2000)	The options that appear in the button popup list, separated by pipe characters "|".
customsql	nvarchar(2000)	The custom SQL to run when the button is clicked.
hotkey	nvarchar(32)	The combination of keys that perform the same function as clicking the button.

VL_ANNOTATIONSETS

The VL_ANNOTATIONSETS table contains the annotations sets created by the user. The following chart describes the columns in the VL_ANNOTATIONSETS table.

Field	Data Type	Description
set_id	integer	A unique identifier for the annotation set.
name	nvarchar(256)	The name of the annotation set as it appears in the Telescope interface.
owner	nvarchar(32)	The user_name of the user who created the clip. Cross-reference to the USERS table.

VL_CLIPS

The VL_CLIPS table contains the user-defined clip information for time-based extended views and the in and out timestamps for the clip. A fixed schema is defined for clip information, with generic fields that may be useful to the user defining the clip. The following chart describes the columns in the VL_CLIPS table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.
in_msec	integer	In-time for the text, in milliseconds, from the beginning of the video.
out_msec	integer	Out-time for the text, in milliseconds, from the beginning of the video.
in_smpte	nchar(12)	In-time for the text, represented in SMPTE time code, if it is encoded in the incoming video stream.
out_smpte	nchar(12)	Out-time for the text, represented in SMPTE time code, if it is encoded in the incoming video stream.
title	nvarchar(255)	*DEPRECATED.*
subject	nvarchar(255)	*DEPRECATED.*
source	nvarchar(255)	*DEPRECATED.*
content_date	timestamp	Date for the clip (this may be the date the clip was taken or some other date useful to the user).
aux_1	nvarchar(255)	Extra field 1.

Field	Data Type	Description
aux_2	nvarchar(255)	Extra field 2.
clips_description	nvarchar(max)	Clip description entered by the user.
clip_id	integer	A number representing the clip.
public_yn	nchar(1)	Indicates whether the clip is public or private.
owner	nvarchar(32)	The user_name of the user who created the clip. Cross-reference to the USERS table.

VL_INFO

The VL_INFO table stores information captured by video processing software. The following chart describes the columns in the VL_INFO table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.
start_offset	integer	The start time, in milliseconds, of a video clip.
end_offset	integer	The end time, in milliseconds, of a video clip.
last_update	datetime (timestamp)	Indicates when the clip was last updated.
frame_rate	real	The video frame rate used for converting to SMPTE timecode.
tape_offset	integer	The difference (in milliseconds) between the beginning of the physical tape and the start of the digital file.
duration	big integer	The length of the video in milliseconds.

VL_PLAYLISTS

The VL_PLAYLISTS table contains the user-defined playlists. Each playlist clip is defined by a record in this table with a unique playlist_id. The following chart describes the columns in the VL_PLAYLISTS table.

Note: If ip_viravideorendition in the DB_SETTINGS table is configured with 'vl_proxies.url' (that is, if the parent asset of the clip has a URL entry in VL_PROXIES table), then the entry created in the VL_PLAYLISTS table is always populated whenever clips are added to a playlist.

Field	Data Type	Description
playlist_id	integer	A unique identifier for the playlist..
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.
in_msec	integer	The in time, in milliseconds, of the playlist clip relative to the playlist asset.
out_msec	integer	The out time, in milliseconds, of the playlist clip relative to the playlist asset.
title	nvarchar(255)	The playlist clip title.
asset_id	integer	The asset_id of the video asset the clip is derived from.
rend_id	integer	The rendition of the video asset the clip is derived from.
url	nvarchar(255)	The proxy URL for the clip. This field is populated when clips are added to a docked playlist. It is copied from the url field of the VL_PROXIES table via triggers.
asset_in_msec	integer	The in time, in milliseconds, of the clip relative to the video asset the clip is derived from.
asset_out_msec	integer	The out time, in milliseconds, of the clip relative to the video asset the clip is derived from.

VL_PROXIES

The VL_PROXIES table stores any streaming video proxies for time-based extended views that are not "physical" files, such as Real Video Server streams, etc. The following chart describes the columns in the VL_PROXIES table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.
mime_type	nvarchar(64)	MIME type of the proxy; for example, "application/mpeg".
url	nvarchar(255)	The URL to get to the video proxy.
pixel_height	integer	Height in pixels of the video proxy.
pixel_width	integer	Width in pixels of the video proxy.
frame_rate	real	Frame rate of the video proxy.

VL_TEXT

The VL_TEXT table contains text tracks extracted from the video for time-based extended views and the in and out timestamps for when the text was extracted. The following chart describes the columns in the VL_TEXT table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.
in_msec	integer	In-time for the text, in milliseconds, from the beginning of the video.
out_msec	integer	Out-time for the text, in milliseconds, from the beginning of the video.
in_smpte	nchar(12)	In-time for the text, represented in SMPTE time code, if it is encoded in the incoming video stream.
out_smpte	nchar(12)	Out-time for the text, represented in SMPTE time code, if it is encoded in the incoming video stream.
track_id	big integer	A decimal value that indicates the type of text track this is. These types are stored in the VL_TRACKS table; refer to that table for a list of possible values.
confidence	integer	Integer between 0 and 100 indicating the degree of confidence in the contents of the text (this is mostly used for auto-generated text where there is a possibility of errors in the recognized text).
text_data	nvarchar(max)	Actual text extracted from the video stream.
lang_id	nchar(10)	Identifies the language (from the language_local table, lang_id column). For example, en_US.

VL_THUMBNAILS

The VL_THUMBNAILS table contains all of the thumbnails extracted from a video for time-based extended views and the in and out timestamps for when the thumbnail appears in the video stream. The following chart describes the columns in the VL_THUMBNAILS table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.

Field	Data Type	Description
in_msec	integer	In-time for the thumbnail, in milliseconds, from the beginning of the video.
out_msec	integer	Out-time for the thumbnail, in milliseconds, from the beginning of the video.
in_smpte	nchar(12)	In-time for the thumbnail, represented in SMPTE time code, if it is encoded in the incoming video stream.
out_smpte	nchar(12)	Out-time for the thumbnail, represented in SMPTE time code, if it is encoded in the incoming video stream.
thumbnail	binary	Actual thumbnail, representing the keyframe. This will always be stored in JPEG format.

VL_TRACKS

The VL_TRACKS table contains the text tracks defined for video assets. Nine tracks are added to the database automatically by the Database Manager (DBManager) application when the database is created or updated. The following chart describes the columns in the VL_TRACKS table.

Field	Data Type	Description
track_id	big integer	A decimal value that represents the possible track type. See the list below.
track_name	nvarchar(32)	The name of the track, as users will see it in the Telescope application. See the list below.

Track ID Values

The following decimal values are defined in the track ID column for both the VL_TEXT and VL_TRACKS tables. They are generated from the hex values of 4-character codes, as shown in the table below.

Decimal value (in the track_ID column)	TSAdmin Name (in the track_name column)	4-character code	Hex value
1667462264	Closed Caption	cctx	0x63637478
1953264760	Teletext	tltx	0x746C7478

1634627183	Annotatio	anno	0x616E6E6F
1635083372	Audio Classification	audl	0x6175646C
1936745320	Speech-to-Text	spch	0x73706368
1936746852	Speaker ID	spid	0x73706964
1717791076	Face Recognition	faid	0x66636964
1868788340	On-Screen OCR	ocrt	0x6F637274
1397117774	Speech-to-text	sfsc	0x5346534E

ZOOM_INFO

The ZOOM_INFO table store the zoom info cache information, which will be loaded by the Zoom Broker when it starts, and updated when the cache changes. The following chart describes the columns in the ZOOM_INFO table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.
rend_id	integer	The rend_id of the image asset.
file_location	nvarchar(2000)	The physical file location for the rendition.
long_name	nvarchar(2000)	Viewable name (path) for the file that is used to populate the popup menu in the Document Info window for file path and for user searches on file path.
file_type	nchar(4)	The type of file that is referred to by this rendition (e.g., "TIFF" or "ESPF"). For display purposes, this field is used to map into the TYPE_CODES table, which provides a "full-text" description of the file types.
file_name	nchar(255)	A simple name of the file without any preceding path.
file_size	big integer	The size, in bytes, of the original file, which can be used to determine download times. This field is a 'bigint' database type, which means that Telescope can accurately represent files whose size is up to 264 bytes (8 million terabytes).

Field	Data Type	Description
file_info	nvarchar(255)	Free-form text filled-in during ingest by the I-Piece that reads the file or by Telescope if the file is a graphic. It contains general information about the file, such as the resolution and color depth for images or the sample size and sample rate for audio assets. This field is displayed as-is to the user in the Editorial View. The format of this information is not standardized and can change from one Telescope version to the next.
create_date	timestamp	Date the physical file was created on the disk. It is not user-editable and contains the actual created date of the physical file. Note: The original system create and modify dates may not always be preserved when a file is imported, because of differences in data handling between different operating systems.
mod_date	timestamp	Date the physical file was last modified (as known when the file was put into the database). There is the danger that this field will get out of date if the user modifies the document outside of Telescope, but it can be resynchronized through the use of the "Synchronize Documents" option. Note: The original system create and modify dates may not always be preserved when a file is imported, because of differences in data handling between different operating systems.
file_checksum	nchar(32)	The hexadecimal representation of the checksum for the file. If the file cannot be found, the entry will be NULL. Checksum information is used to link placed art back to the parent document. Note: This checksum information is customized by Telescope and cannot be compared with generated by other tools.
xmp_docid	nvarchar(36)	If Adobe InDesign files are imported using an older version of the Xinet plugin, the checksum information may not be populated into the file_checksum field. Instead, similar document identification information is stored in this xmp_docid field.
page_no	integer	The page number within the assets that is zoomed. This is applicable to PDF files only.

Functional Rules Tables

The Telescope Functional Rules engine is a powerful feature that gives Telescope Administrators an almost unlimited ability to customize the operation of their Telescope installations to meet the business requirements of their organization. Functional Rules are a complex, scripting-based feature, and the intent of this document is not to explain Functional Rules in detail, but to describe how they are stored and managed within the Telescope database itself. See the *Telescope – Administrator's Reference Manual* for more information about Functional Rules in Telescope.

FN_MESSAGES

The FN_MESSAGES table contains the text of error messages that are returned from the execution of a rule that fails. It associates function return codes with messages that are returned to users. The following chart describes the columns in the FN_MESSAGES table.

Field	Data Type	Description
fm_id	integer	A unique key for this table.
rule_id	integer	The ID of the rule to which this message pertains.
result_code	integer	The result code which will cause this message to be generated. A result code of 0 (zero) in this field indicates the "default" error message that will be returned for any functional return codes not explicitly listed for the rule. Use a number below -10. (-1 through -10 are reserved, and positive numbers are not used for error messages.)
err_text	nvarchar(255)	The text of the error message to be returned to the user. This text can contain parameter substitutions as well as actual text.

FN_RULES

The FN_RULES table stores the information about the functional rules defined for the system. The following chart describes the columns in the FN_RULES table.

Field	Data Type	Description
rule_id	integer	The unique ruleID associated with this functional rule.

Field	Data Type	Description
rule_name	nvarchar(255)	The name of the rule as it gets displayed to the administrator. Tip: This text may appear in a menu, so do use a descriptive name.
test_func	nvarchar(2048)	The SQL script to be executed as the test function, including parameter replacement tags.
challenge_form	nvarchar(4000)	The XML representation of the challenge form, including parameter replacement tags. If this column is empty, then there is no challenge form, and the RESPONSE_FUNC column will be ignored by Telescope.
response_func	nvarchar(2048)	The SQL script to be executed as the response function, including parameter replacement tags. If the CHALLENGE_FORM column is empty, then this column's value is ignored by Telescope.

FN_RULESETS

The FN_RULESETS table associates sets of rules with particular user groups, and particular actions. The following chart describes the columns in the FN_RULESETS table.

Field	Data Type	Description
fr_id	integer	A unique key for this table. This is an administrative ID, generated by the Telescope Admin application.
group_name	nvarchar(32)	The user group to which this group applies. This is a reference to the USERS.USER_NAME field for a user group.

Field	Data Type	Description
action_code	integer	A numeric code representing the action to which this rule should be applied. Valid values for this field are: **1** Copy **2** Metadata Update **3** Delete **4** Import (pre-flight). This action code was simply "Import" prior to Version 9.3.1, and typically was executed post-flight (except for the Telescope Uploader, which executed this action pre-flight). **6** Move **7** Drag (DEPRECATED) **8** Check out **9** Check in **10** Update **11** Attach Rendition **12** Open Extended View **13** Order **14** Conversions **15** Add to Collection (Catalog) **16** Remove from Collections (Catalog) **17** Menu Rule **18** Login **19** QuickLinks **20** Access Collections (Catalogs) **41** Import (post-flight).
ruleset_order	integer	An integer indicating the ordering of the rule in the ruleset for this group, for this action.
rule_id	integer	A reference back to the FN_RULES table for the rule that should be applied at this position in the ruleset.

FN_WATERMARKS

Whether a watermark is displayed is controlled by Functional Rules triggered by the "Extended View'" event. The result of the Test Function should be a watermark_id from the FM_WATERMARKS table. The associated image becomes the watermark. The following chart describes the columns in the FN_WATERMARKS table.

Field	Data Type	Description
watermark_id	integer	A positive integer that matches a return code from the "view extended view" functional rule.

Field	Data Type	Description
watermark_name	nvarchar(255)	A text description field for the watermark. Telescope doesn't use this field directly, but it would be useful as a lookup from the Functional Rule itself, so that the ID of the watermark doesn't need to be hard-coded in the Functional Rule.
watermark_image	binary	The image to be used as the watermark. This is a binary column whose data must be in JPEG, PNG or GIF format. To be appropriately overlaid over the extended view image, the watermark image should have an alpha channel, which requires either PNG or GIF formats.

Order Entry Tables

Telescope provides a complete and flexible order entry system. See the "Fulfillers" chapter in the *Telescope Administrator's Reference Manual* for details on how to administer and maintain order entries.

The following tables manage and store data about the structure of the order entry system, and the orders placed by users.

EXT_ADDRESSES

The EXT_ADDRESSES table contains "address book" entries used by the Order Entry component of Telescope. This information is required for orders that are physically shipped to their recipients. Each user may have and maintain their own address book. The following chart describes the columns in the EXT_ADDRESSES table.

Field	Data Type	Description
id	integer	A unique key for this table, automatically generated.
user_name	nvarchar(32)	The user for which this address book entry applies.
addr_name	nvarchar(32)	The user's "helpful name" for this address, for example, "Home" or "Work".
contact_name	nvarchar(255)	Address information.
company_name	nvarchar(128)	Address information.
department	nvarchar(128)	Address information.
address_1	nvarchar(255)	Address information.
address_2	nvarchar(255)	Address information.
city	nvarchar(64)	Address information.
state	nvarchar(64)	Address information.
country	nvarchar(64)	Address information.
postzip	nvarchar(32)	Address information.
phone_number	nvarchar(32)	Address information.
email	nvarchar(128)	Address information.

OE_ARCHIVE

The OE_ARCHIVE table contains the archived copies of orders that are moved out of the order processing table structure by the Auto-Archive process. The following chart describes the columns in the OE_ARCHIVE table.

Field	Data Type	Description
archive_id	integer	A unique key for this table. This is a "transparent" identifier for Telescope that is populated automatically when a record is inserted into this table. It is a true "identity" column in SQL Server, and an integer populated by a sequence and an insert trigger in Oracle.
from_category	nvarchar(64)	The name (i.e. the fulfiller_category field from the OE_FULFILLERS table) of the fulfiller category to which the fulfiller of this order belongs. Over time, users may be moved from one fulfiller category to another, be "demoted" from being a fulfiller, or be deleted from the database entirely. In this case, it is useful to know what fulfiller category the fulfiller user (whose user_name is captured in the order archive XML) belonged to at the time the order was archived.
archive_date	datetime (timestamp)	The date and time the order was removed from the order processing tables and added to the archive.
order_text	nvarchar(max)	XML formatted text that contains all of the information about the order at the time it was moved to the archive.

OE_ASSETMETADATA

The OE_ASSETMETADATA table contains the metadata information for a specific order against a specific asset within that order. Order forms may have an unlimited number of form fields in their HTML definition. Any additional fields on the form that are not otherwise used by the order entry system are gathered by Telescope as order 'metadata', and stored in the OE_ASSETMETADATA table for reference. The following chart describes the columns in the OE_ASSETMETADATA table.

Field	Data Type	Description
order_id	integer	The ID of the order to which this metadata belongs.
position	integer	The position (1 to n) of the metadata field on the form. This is used to provide consistency in the display of the form to the user after the form has been submitted.

Field	Data Type	Description
fieldname	nvarchar(255)	The name of the form field from which the data comes. If the field on the form has multiple values (for example, a multi-select list), then there will be multiple instances of the same fieldname record in the table for this order. The position value then determines which order the individual items are listed within the specific field. When the form designer designs the HTML order form, they should give the field names which take into account the fact that they will be converted to a "human friendly" format by replacing underscores with spaces and then capitalizing the first letter of each word. So, "shipping_mode" is a good field name (as it will be translated into "Shipping Mode"), whereas "shmod" is not, since it will be translated into "Shmod".
value	nvarchar(255)	The value of the field.

OE_ASSETOUTVALS

The OE_ASSETOUTVALS table lists possible output formats for each asset (Transparency, etc.). The following chart describes the columns in the OE_ASSETOUTVALS table.

Field	Data Type	Description
fulfiller_id	integer	The fulfiller for whom the order output values apply.
val_id	integer	A value (1 to n) where n is the number of values for a given fulfiller. Used to order the values in the menu.
valuestr	nvarchar(64)	Description of the output format.
quantity_yn	nchar(1)	A flag that indicates whether this output format requires a quantity or not. For example, "Transparency" would require a value separate from the rest of the order, because the user could want different numbers of items for each transparency they order. However, "Electronic" for example, would not require a quantity since it is an electronic delivery method, and specifying a quantity doesn't make much sense. When the user fills out the order form, if they select any asset output formats for which quantity_yn is "N", then the OrderFormat component will appear on the order form, so the user can specify an overall order number and format (for example "2" and "CD" to indicate how the "Electronic" files are to be delivered).

OE_ASSETS

The OE_ASSETS table lists all assets associated with an order. The following chart describes the columns in the OE_ASSETS table.

Field	Data Type	Description
order_id	integer	The ID of the order to which the asset belongs.
position	integer	The position (1 to n) of the item in the order.
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table being ordered.
rend_id	integer	The rendition ID of the ordered file.
asset_status	nvarchar(255)	The status of the ordered asset. When the order is first created, this is set to the "initial" value from the OE_ASSETSTATVALS table. In the event of an error converting or copying the asset to the fulfiller's fulfillment location, this status will be set to the "error" value from the OE_ASSETSTATVALS table.
conv_str	nvarchar(4000)	The conversion string selected for the file, if any. This is a File Conversion Broker formatted conversion string.
quantity	integer	The quantity ordered.
out_format	nvarchar(255)	The output format selected for the ordered file (values come from the OE_ASSETOUTVALS table).
final_location	nvarchar(512)	A human-readable path to the final location of the copied and converted file.
version_id	integer	The version ID of the asset required. Cross-reference to ed_versions.version_id.

OE_ASSETSTATVALS

The OE_ASSETSTATVALS table lists all possible status values for each asset. The following chart describes the columns in the OE_ASSETSTATVALS table.

Field	Data Type	Description
fulfiller_id	integer	The fulfiller for whom the asset output values apply.
val_id	integer	A value (1 to n) where n is the number of values for a given fulfiller. Used to order the values in the menu.
valuestr	nvarchar(64)	Description of the status.

Field	Data Type	Description
error_yn	nchar(1)	Whether this is the "error" value for the asset's status value
initial_yn	nchar(1)	Whether this is the initial value for the asset's status value.

OE_FULFILLERS

Any user in the Telescope system can be made a fulfiller, which indicates that they are notified when orders are placed, and are responsible for fulfilling orders placed by other users. When a user places an order, they choose a fulfiller to handle the order. Fulfillers are divided into multiple fulfiller 'categories', which can have different properties, such as order forms, etc. The following chart describes the columns in the OE_FULFILLERS table.

Field	Data Type	Description
fulfiller_id	integer	A unique key for this table, automatically generated.
fulfiller_category	nvarchar(64)	The user-friendly name of the "category" of the fulfiller. This is useful if there are multiple user records which share a single OE_FULFILLER record, in which case this name will group the fulfiller users together in the fulfiller popup menu in the event that a user can see multiple fulfillers in the same category.
order_broker	nvarchar(64)	The name of the File Broker where the ordered assets will be placed when a user orders files, and chooses this user as the fulfiller. If left blank, no order preparation is done by the system (i.e. manual order preparation).
order_share	nvarchar(64)	The name of the share on order_broker (above) where the order assets will be placed when a user orders files, and chooses this user as the fulfiller.
order_html	nvarchar(64)	The name of the HTML file that will be used for this fulfiller's order form. This value can be left empty to specify the default order form.
category_submit	nchar(10)	Indicates what an ordering user will see in the Fulfillers list when they place an order. Valid values for this field are: **Off** – ordering users will see a list of user names in the Fulfillers list. **On** – ordering users will see both user names and Fulfiller Category names in the Fulfillers list. **Require** – ordering users will see only Fulfiller Categories in the Fulfillers list.

OE_METADATA

The Order Fulfillment Module (OFM) allows for metadata within the order to be tied either to the order (overall) or to individual assets within the order. The OE_METADATA table contains metadata information that was entered that is specific to the overall order. The following chart describes the columns in the OE_METADATA table.

Field	Data Type	Description
order_id	integer	The ID of the order to which this metadata belongs.
position	integer	The position (1 to n) of the metadata field on the form. This is used to provide consistency in the display of the form to the user after the form has been submitted.
fieldname	nvarchar(255)	The name of the form field from which the data comes. If the field on the form has multiple values (for example, a multi-select list), then there will be multiple instances of the same fieldname record in the table for this order. The position value then determines which order the individual items are listed within the specific field. When the form designer designs the HTML order form, they should give the field names which take into account the fact that they will be converted to a "human friendly" format by replacing underscores with spaces and then capitalizing the first letter of each word. So, "shipping_mode" is a good field name (as it will be translated into "Shipping Mode"), whereas "shmod" is not, since it will be translated into "Shmod".
value	nvarchar(255)	The value of the field.

OE_ORDERS

The OE_ORDERS table is the base order entry table. For each order placed in the system, there is an entry in the OE_ORDERS table. The following chart describes the columns in the OE_ORDERS table.

Field	Data Type	Description
order_id	integer	A unique key for this table that is generated automatically for the order.
user_name	nvarchar(32)	The user who placed the order.
order_status	nchar(32)	The status of the order. The possible values in this field will come from the OE_STATVALS table.
date_placed	timestamp	The date and time the order was originally placed by the user.

Field	Data Type	Description
date_fulfilled	timestamp	The date and time the order was "fulfilled" by the fulfiller. This is the date and time that the order's ORDER_STATUS value was set to the "final" value by the fulfiller. If the order has never had its ORDER_STATUS value set to the "final" value by the fulfiller, then this column contains NULL.
last_modified	timestamp	The date and time the order was last modified, by anyone.
last_assigned	timestamp	The date and time the order's fulfiller was changed. When the order is initially placed, this column contains the same date and time as date_placed.
contact_name	nvarchar(255)	Address information.
company_name	nvarchar(128)	Address information.
department	nvarchar(128)	Address information.
address_1	nvarchar(255)	Address information.
address_2	nvarchar(255)	Address information.
city	nvarchar(64)	Address information.
state	nvarchar(64)	Address information.
country	nvarchar(64)	Address information.
postzip	nvarchar(32)	Address information.
phone_number	nvarchar(32)	Address information.
email	nvarchar(128)	Address information.
order_quantity	integer	Quantity for this order.
order_outformat	nchar(255)	Output format for this order.
fulfiller_user	nvarchar(32)	The user who is the fulfiller for this order.

OE_OUTVALS

The OE_OUTVALS table lists possible output formats for the entire order (CD, DVD, ZIP, etc.) The following chart describes the columns in the OE_OUTVALS table.

Field	Data Type	Description
fulfiller_id	integer	The fulfiller the order output values apply to.

Field	Data Type	Description
val_id	integer	A value (1 to n) where n is the number of values for a given fulfiller. Used to order the values in the menu.
valuestr	nvarchar(64)	Description of the output format.

OE_STATVALS

The OE_STATVALS table lists all possible values for the order status. The following chart describes the columns in the OE_STATVALS table.

Field	Data Type	Description
fulfiller_id	integer	The fulfiller the asset output values apply to.
val_id	integer	A value (1 to n) where n is the number of status values for a given fulfiller used to order the values in the menu.
value_text	nvarchar(64)	The status value.
initial_yn	nchar(1)	Whether this is the "initial" value for the order status when it is placed.
final_yn	nchar(1)	Whether this is the "final" or "closed" value for the order status.
trigger_script	nvarchar(255)	SQL script to execute when the order's status changes to this value. In the text of this script, the token "<!order_id!>" will be replaced with the order_id of the order.

User Tables

The Telescope USERS table stores information about each user and group in the system. It is used to authenticate users by their username and password values when they log in. The Authentication Broker can be configured to use other authentication methods, such as LDAP. The users and group permissions are also stored in the USERS table.

Other tables used to define users include:

- DOWNLOAD_QUEUE: stores the user's download cart for Telescope
- EXTENDEDVIEW_FIELDS: PARAVIEW_FIELDS, TEXTVIEW_FIELDS and TNAILVIEW_FIELDS: store user preferences about which metadata fields they wish to view in the various Telescope display modes.
- VIEW_... tables: This set of tables define users and group visibility privileges (i.e. which system objects, such as searches, metadata fields, etc., users have permission to see).

ANNOUNCEMENT_LIST_GROUPS

The ANNOUNCEMENT_LIST_GROUPS table stores the names of user groups and the announcement lists to which they are subscribed to. The following chart describes the columns in the ANNOUNCEMENT_LIST_GROUPS table.

Field	Data Type	Description
group_name	nvarchar(32)	User name of the group for whom subscribing to the associated announcement list. Cross-reference to the user_name field in the USERS table.
list_id	integer	The ID for the announcement list. Cross-reference to the ANNOUNCEMENT_LISTS table.

ANNOUNCEMENT_LIST_MODERATORS

The ANNOUNCEMENT_LIST_MODERATORS table stores the names of user groups and the announcement lists to which they are subscribed. The following chart describes the columns in the ANNOUNCEMENT_LIST_MODERATORS table.

Field	Data Type	Description

Field	Data Type	Description
user_name	nvarchar(32)	User name of the user with moderator rights (add, change, delete, send to users) for a particular list. Cross-reference to the user_name field in the USERS table.
list_id	integer	The ID for the announcement list. Cross-reference to the ANNOUNCEMENT_LISTS table.

ANNOUNCEMENT_LISTS

The ANNOUNCEMENT_LISTS table stores the names of announcement lists. The following chart describes the columns in the ANNOUNCEMENT_LISTS table.

Field	Data Type	Description
list_id	integer	The ID for the announcement list.
list_name	nvarchar(32)	Convenience name of the announcement list.

ANNOUNCEMENTS

The ANNOUNCEMENTS table stores the information about individual announcements. The following chart describes the columns in the ANNOUNCEMENTS table.

Field	Data Type	Description
announcement_id	integer	The ID for the announcement.
list_id	integer	The ID for the announcement list. Cross-reference to the ANNOUNCEMENT_LISTS table.
sent_date	datetime (timestamp)	The date and time when the announcement is initially broadcast. This is referenced to allow Telescope to determine which announcements a logged-in user has not seen, and mark them as "New".
sending_user	nvarchar(32)	The user name of the creator of this announcement.
attached_catalog	integer	ID of a collection (catalog) to be sent with this announcement.

Field	Data Type	Description
attached_action	nvarchar(10)	This field determines how to view the attached collection (catalog). Possible values are: **catalog** – Displays all assets in the collection. **Details** – Display the document information for the first asset in the collection. **Preview** – Displays the extended view for the first asset in the collection.
announcement_title	nvarchar(255)	The subject of the announcement which will display above/before the announcement_text.
announcement_text	nvarchar(4000)	The body of the message. HTML tags are allowed.
lang_id	nchar(10)	Identifies the language (from the language_local table, lang_id column). For example, en_US.

DOWNLOAD_QUEUE

The DOWNLOAD_QUEUE table lists all of the documents that the user has selected for download using Telescope. This list is preserved between logins for users.

There may be environments where it is known that a specific user under specific circumstances will download an asset. In order to streamline workflow, a customization could be created to automatically populate that user's download basket and send the user an email indicating that the asset is ready for download. This would be useful in an environment where a user "orders" an asset and has to wait for the request to be processed.

The following chart describes the columns in the DOWNLOAD_QUEUE table.

Field	Data Type	Description
id	identity	Unique ID generated automatically on insert to the table. On Oracle, this is a plain integer column with an insert trigger to populate it from an Oracle sequence.
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
user_name	nvarchar(32)	The user who initiated the download or added the asset to the download queue.
rendition	integer	Cross-reference to the DOC_RENDITIONS table, indicating the rendition of the document that the user has chosen to download.

Field	Data Type	Description
status	integer	Code representing the status of the download operation. Valid values for this field are: **1** Pending download **2** In Progress **3** Error **4** Pending Approval **5** Pending Checkout
errmsg	nvarchar(255)	If a document does not download successfully, its status value will be set to "3", and its errmsg field will contain a descriptive error message.
conv_string	nvarchar(max)	A conversion string the user defined for the file to be downloaded. This string is sent verbatim to the FBOpenConvert call of the File Broker. It is also displayed in the download basket next to the convert button. If this field is empty, no conversion is applied to the downloaded file and the following field is not used. Note that a chain conversion is allowed here. Chain conversion as separated by the pipe character "|".
conv_broker	nvarchar(255)	Name of the File Broker that can perform the above conversion. Instead of locating a File Broker to download the file from in the normal way, Telescope uses this specific File Broker for this file.
copycov_yn	nchar(1)	This flag is available for COV documents only. A value of "Y" in this field indicates that the user has chosen to download the placed documents as well.
actor_name	nvarchar(60)	The user who actually performed the download and the name that will be added to the access history table. Primarily this is for functionality such as QuickLinks, where the user performing the download is not a user on the Telescope database.
fr_conv_string	nvarchar(max)	When the "copy files" functional rule is executed, it may return a conversion string which is "forced" onto the file at download time. Because the functional rule is executed by Telescope and the conversion is executed later by the Download Manager, the conversion string returned by the functional rule must be stored here in the download_queue table to pass to the Download Manager. This string can contain a "chained" conversion, delimited by the pipe "|" character and it should be appended to the end of the string contained in conv_string, if any. This string will begin with a pipe "|" character so that, when appended to the conv_string, it will create a proper chained conversion string.

Field	Data Type	Description
version_id	integer	Cross-reference to the ED_VERSIONS table for this version.

EXTENDEDVIEW_FIELDS

The EXTENDEDVIEW_FIELDS table stores user preferences about which fields users want to see on each preview in the extended view. The following chart describes the columns in columns in the EXTENDEDVIEW_FIELDS table.

Field	Data Type	Description
user_name	nvarchar(32)	User name of the user for whom this preference applies. This value refers back to the user_name field in the USERS table.
first_field	nvarchar(64)	The name of the first field from the metadata model to show in the extended view. This is the column_name field in the EXTRA_COLUMNS table.
second_field	nvarchar(64)	Same as above, second field.
third_field	nvarchar(64)	Same as above, third field.
fourth_field	nvarchar(64)	Same as above, fourth field.
rendition	integer	Default rendition (from DOC_RENDITIONS) that is displayed in elements of this view.

PARAVIEW_FIELDS

The PARAVIEW_FIELDS table stores user preferences about which fields users want to see in the paragraph view. The following chart describes the columns in the PARAVIEW_FIELDS table.

Field	Data Type	Description
user_name	nvarchar(32)	User name of the user for whom this preference applies. Cross-reference to the user_name field in the USERS table.
first_field	nvarchar(64)	The name of the first field from the metadata model to show in the paragraph view. This is the column_name field in the EXTRA_COLUMNS table.
second_field	nvarchar(64)	Same as above, second field.
third_field	nvarchar(64)	Same as above, third field.
fourth_field	nvarchar(64)	Same as above, fourth field.

Field	Data Type	Description
fifth_field	nvarchar(64)	Same as above, fifth field.
sixth_field	nvarchar(64)	Same as above, sixth field.
seventh_field	nvarchar(64)	Same as above, seventh field.
eighth_field	nvarchar(64)	Same as above, eighth field.
rendition	integer	Default rendition (from DOC_RENDITIONS) that is displayed in elements of this view.

QL_ASSETS

The QL_ASSETS table stores lists of assets to be sent via QuickLinks. The following chart describes the columns in the QL_ASSETS table.

Field	Data Type	Description
ticket_id	integer	The ID for the Quick Link. See QL_TICKETS table.
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table for the asset being sent.
rend_id	integer	Cross reference to the rend_id field in the DOC_RENDITIONS table.
conv_str	nvarchar(max)	The conversion string to be applied to this asset before offering to the user for download.

QL_RECIPIENTS

The QL_RECIPIENTS table stores the email address of the recipients of the QuickLinks queued in the system. The following chart describes the columns in the QL_RECIPIENTS table.

Field	Data Type	Description
ticket_id	integer	The ID for the QuickLinks. See QL_TICKETS table.
recipient	nvarchar(255)	The e-mail address of the QuickLink to be sent. Each row in this table holds a single address.
access_key	nvarchar(8)	A randomly generated security key that will be sent to the user in a separate email message. This key is required to access the asset(s) for download.

QL_TICKETS

The QL_TICKETS table stores the lists of assets to be sent via QuickLinks. The following chart describes the columns in the QL_TICKETS table.

Field	Data Type	Description
ticket_id	integer	The ID for the QuickLink.
sending_user	nvarchar(32)	The user_name of the creator of the referenced QuickLink. Cross-reference to user_name in the USERS table.
sent_date	datetime (timestamp)	Date and time when the QuickLink was sent.
expiry_date	datetime (timestamp)	Date and time when this ticket is invalid (i.e., not available for download).
expire_after	integer	Number of hours after the send_date to invalidate this ticket.
dl_remaining	integer	Number of downloads remaining before this ticket is invalid.

TEXTVIEW_FIELDS

The TEXTVIEW_FIELDS table stores user preferences about which fields users want to see on each line of text in the text view. The following chart describes the columns in the TEXTVIEW_FIELDS table.

Field	Data Type	Description
user_name	nvarchar(32)	User name of the user for whom this preference applies. This value refers back to the user_name field in the USERS table.
first_field	nvarchar(64)	The name of the first field from the metadata model to show in the text view. This is the column_name field in the EXTRA_COLUMNS table.
second_field	nvarchar(64)	Same as above, second field.
third_field	nvarchar(64)	Same as above, third field.
fourth_field	nvarchar(64)	Same as above, fourth field.
fifth_field	nvarchar(64)	Same as above, fifth field.
sixth_field	nvarchar(64)	Same as above, sixth field.
seventh_field	nvarchar(64)	Same as above, seventh field.

Field	Data Type	Description
eighth_field	nvarchar(64)	Same as above, eighth field.
rendition	integer	Default rendition (from DOC_RENDITIONS) that is displayed in elements of this view.

TNAILVIEW_FIELDS

The TNAILVIEW_FIELDS table stores user preferences about which fields they want to see under each thumbnail in the thumbnail view. The following chart describes the columns in the TNAILVIEW_FIELDS table.

Field	Data Type	Description
user_name	nvarchar(32)	The name of the user for whom this preference applies. This value refers back to the user_name field in the USERS table.
first_field	nvarchar(64)	The name of the first field from the metadata model to show in the thumbnail view. This is the column_name field in the EXTRA_COLUMNS table.
second_field	nvarchar(64)	Same as above, second field.
third_field	nvarchar(64)	Same as above, third field.
fourth_field	nvarchar(64)	Same as above, fourth field.
fifth_field	nvarchar(64)	Same as above, fifth field.
sixth_field	nvarchar(64)	Same as above, sixth field.
rendition	integer	Default rendition (from DOC_RENDITIONS) that is displayed in elements of this view.

UPLOAD_QUEUE (Deprecated)

Deprecated.

USERS

Telescope is access-controlled software. Users must log in to the database, and their activities are controlled by access privileges. The USERS table details the access privileges for each user, as well as personal information, such as name and contact information.

The USERS table is very useful for creating customizations that require external communications such as email. For example, you could create a customization to monitor assets in a Telescope environment and alert a group or division leader by email any time an asset is assigned to the group.

The following chart describes the columns in the USERS table.

Field	Data Type	Description
userid	integer	*DEPRECATED.*
user_name	nvarchar(32)	The login name of the user. This is the unique key for this table.
password	nvarchar(32)	The password for the user. This field is encrypted in the database to prevent unauthorized access.
access_where	nvarchar(2000)	For Tree Searches, the portion of an SQL where clause to be appended to each query Telescope generates. This can be used to restrict access to particular records in the database. For example, if the access_where field for a user contains "category = 'SPT' or category = 'WLD'", and the user causes Telescope to generate a query like: "Select ... from editorial where country = 'Canada'", the final query submitted to the database engine would be: "Select ... from editorial where country = 'Canada' and category = 'SPT' or category = 'WLD'". Note: This field is for the traditional SQL search method, which is not recommended. For Solr search, refer to the access_where_solr field.
access_flags	nchar(20)	Interpreted by Telescope as a sequence of individual characters, where each character represents access permission for functionality. A "Y" in a particular position indicates that the position is "set" or that the user has the given privilege. The character positions in the field are defined as follows: 1 If set, the user is an administrator and can use Telescope Administrator to set administrative options for the database. 2 *DEPRECATED*. 3 If set, the user can download documents from the database. 4 If set, the user can view files using the scrolling Thumbnail View. If clear, the user can only use the Text View for viewing files. This field is useful for remote access users with limited bandwidth. 5 If set, the user can import files into the database using Telescope. 6 If set, the user can see the extended view of the documents in the database.

Field	Data Type	Description
		7 If set, the user can delete files from the database. 8 If both this position and position 7 are set, the user must obtain an approval before deleting file(s) from the database. 9 If both this position and position 3 are set, the user must obtain an approval before downloading file(s) from the database. 10 If set, the user may issue approvals for deletions. 11 If set, the user may issue approvals for downloads. 12 If set, the user may change his or her own password. If this field is not set, then the user must ask the database administrator to change the password. 13 If set, the user may use the "Change Multiple" command in Telescope. 14 *DEPRECATED.* 15 If set, the user may use the "Move Files" command in Telescope. 16 If set, the user may add and remove jobs (templates) from the Jobs table. 17 *DEPRECATED.* 18 If set, the user may create public collections (catalogs). If not set, the user may only create private collections.
max_results	integer	A value that limits the number of hits a user can successfully have on the results of an SQL query. If this number is exceeded, Telescope issues a warning alert and the user is either restricted to viewing the first max_results records retrieved or placed back into the find dialog. Note: This field is not as relevant for 9.2 or later because of new paging and Solr search architecture. However, it is still useful to prevent timeouts when using Integration Broker SOAP API queries.
grpcrt	nchar(1)	*DEPRECATED.*
grpchg	nchar(1)	*DEPRECATED.*
grpdel	nchar(1)	*DEPRECATED.*
chkout	nchar(1)	Y/N flag (with NULL equating to N) that indicates whether a user is permitted to checkout files from the database.

Field	Data Type	Description
seevers	nchar(1)	Y/N flag (with NULL equating to N) that indicates whether a user is permitted to see previous versions of files in the database or is limited to only viewing current versions.
group_yn	nchar(1)	The Users table keeps records of users and groups in Telescope. This field is a Y/N flag that indicates if the entry is a Group or a User. "Y" in this field indicates that this entry is a Group name, and any other value (including NULL) indicates that this entry is a User.
member_of	nvarchar(32)	Name of the group to which the user belongs. For Group users, this field will be NULL. This is cross-reference to the user_name field of the Group user in the USERS table.
phonenumber	nvarchar(40)	Optional descriptive information about the user.
email	nvarchar(60)	Optional descriptive information about the user.
fname	nvarchar(40)	Optional descriptive information about the user. This is the user's First Name.
lname	nvarchar(40)	Optional descriptive information about the user. This is the user's Last Name.
company	nvarchar(40)	Optional descriptive information about the user.
department	nvarchar(80)	Optional descriptive information about the user.
address	nvarchar(80)	Optional descriptive information about the user.
city	nvarchar(40)	Optional descriptive information about the user.
state	nvarchar(2)	Optional descriptive information about the user.
zipcode	nvarchar(9)	Optional descriptive information about the user.
country	nvarchar(40)	Optional descriptive information about the user.
impapprov	nchar(1)	Y/N flag for "Import With Approval". If set to "Y", the user has this privilege. If it is anything else (including NULL), the user does not have this privilege.
approvimp	nchar(1)	Y/N flag for "Approve Imports". If set to "Y", the user has this privilege. If it is anything else (including NULL), the user does not have this privilege.
appmsgto	nvarchar(32)	Name of the user to whom approvals messages are to be sent, if any. If this field is an empty string or NULL, the user is allowed to choose to whom approval messages go.

Field	Data Type	Description
lastlogin	timestamp	Date and time the user last logged into the Telescope system. Before the user logs in the first time, this value is NULL.
defaultview	short integer	User's default view preference, used by Telescope. This field could be "1" for Thumbnail View, "2" for Text View or "3" for Paragraph View.
usrclass	nchar(2)	User's designation in the two-tiered user model: **CU** – Content Editors (Concurrent Users). **CC** – Content Consumers (Unlimited Browse and Download by Web Users).
remarks	nvarchar(max)	Optional descriptive information about the user.
fulfiller_id	integer	If the user is a fulfiller, the ID in the oe_fulfillers table of the fulfiller record will be entered for this user. Multiple users can be linked to the same oe_fulfillers record. If the user is not a fulfiller, this column will contain NULL.
dl_limit	big integer	The maximum size, in bytes, the user is able to download in one click. If this value is zero, it means that the user can download as much as they want. The maximum value for the download limit is 2Gb.
freeconv_yn	nchar(1)	A permission flag indicating whether the user can use the "standard" conversion functionality on the web. If this field is set to 'N' or NULL, the user can only use the named conversions provided for their group. If this field is set to 'Y', the user may use any named conversions, in addition to the standard "free-form" conversion functionality provided by the File Conversion Broker.
order_yn	nchar(1)	A permission flag indicating whether the user can use the order processing functions.
pool_name	nvarchar(32)	The name of the Session Broker license pool (if any) to which the user belongs.
nofmimp_yn	nchar(1)	This setting determines whether file migration policies apply during an import. If set to Yes, file migration policies apply. If set to No, file migration policies do not apply.
syncdocs_yn	nchar(1)	This field is used to indicate whether the user may use the "Synchronize Documents" command. If set to 'Y', the command is available to the user. If set to 'N', the command is not available to the user.

Field	Data Type	Description
lctdocfiles_yn	nchar(1)	This field is used to indicate whether the user may use the "Locate Document Files" command. If set to 'Y', the command is available to the user. If set to 'N', the command is not available to the user.
dupfiledetection	integer	This field is used to determine how collision detection is handled, according to the following values: **1** File Name **2** File Name and Created Date **3** File Path **4** MD5 *(that is, checksum)*
dupfileresolution	integer	This field is used to determine how duplicate files are handled, according to the following values: **1** Update Existing **2** Insert New **3** Skip File
ovrfiles_yn	nchar(1)	Y/N flag that indicates if the user has permission to overwrite files. If the value is "Y", the user is granted permission to overwrite files. If the value is "N", the user is not granted permission to overwrite files. "N" is the default setting. If overfiles_yn is NULL, the user is granted permission to overwrite files.
tpp_promoteadmin_yn	nchar(1)	If set to 'Y', the user has access to the TPP (Telescope Publishing Platform) Promote application.
tpp_distributeadmin_yn	nchar(1)	If set to 'Y', the user has access to the TPP Distribute application.
findall_yn	nchar(1)	If set to 'Y', the user can see the Find ALL icon/command at the top panel.
checkin_rem_yn	nchar (1)	If set to "Y", the users can opt to remove files from their local drives as they check them in.
hidefmimp_yn	nchar (1)	If set to "Y", the user will not see Migration Policies selection drop down box on the Import page (it will be hidden). If the value is "N", users will be able to see the drop down box.
lang_id	nchar(10)	Identifies the language (from the language_local table, lang_id column) of the user or group. For example, en_US.
translator_yn	nchar (1)	This field is intended for future use.

Field	Data Type	Description
access_where_solr	nvarchar(2000)	The Solr where clause to be appended to each query Telescope generates. This can be used to restrict access for the user to particular records in the database. The Solr where clause is generated in the TSAdmin Users/Groups tab. It is recommended that you create this content through the TSAdmin interface. If you do attempt to update it, ensure that you use uppercase for operators (AND, OR).
uploader_yn	nchar(1)	If set to "Y", users can download and use the Telescope Uploader. If the value is set to "N", users are not allowed to download the Telescope Uploader.
hidequicklinks_yn	nchar(1)	If set to "Y", users can use QuickLinks. If set to "N", they cannot.
plugindownload_yn	nchar(1)	This setting is obsolete with the new Uploader introduced in Telescope 9.4.0.x. If set to "Y", users can use the advanced download plug-in (recommended). When this permission is not checked, users will not be able to use the enhanced download features available with this plug-in, and will not see the "Improve your download experience" link in the Download Cart.
candeleteadvsearch_yn	nchar(1)	If set to "Y", users can delete advanced searches they've made available to the public.
cansaveadvsearch_yn	nchar(1)	If set to "Y", users can create and save public advanced searches Without this permission, users can only create private advanced searches.
date_created	datetime (timestamp)	Provides the date for when the user was added to the table.
assetdock_yn	nchar(1)	If set to "Y", users can use the "Drag and Drop" (Asset Dock) feature.
login_fail_cnt	integer	The count of failed login attempts. Initial default value is 0, incremented by 1 after each failed login attempt. This value is reset to 0 in any of the following situations: • The TSWeb user successfully logs in with the correct password • The administrator unlocks the user in TSAdmin (Users/Groups) • The administrator resets the password

VIEW_ACTIONS

The VIEW_ACTIONS table stores the list of message actions available to each user. The following chart describes the columns in the VIEW_ACTIONS table.

Field	Data Type	Description
user_name	nvarchar(32)	User name. Cross-reference to USERS.user_name.
action_code	nvarchar(64)	The name of the message action available to this user. Cross-reference to the M_MSGACTIONS table.

VIEW_CATALOGS

The VIEW_CATALOGS table stores the list of collection (catalog) permissions available to each user. The following chart describes the columns in the VIEW_CATALOGS table.

Field	Data Type	Description
user_name	nvarchar(32)	The username. Cross-reference to USERS.user_name.
lb_id	integer	ID of the collections in which this item belongs. Cross-reference to the ID column in the M_LIGHTBOXES table.
edit_yn	nchar(1)	If this value is "Y" this user has the right to edit the referenced collection. Any other value, including NULL, prohibits this action.
nest_yn	nchar(1)	If this value is "Y" this user has the right to create collections beneath this one (sub-collections). Any other value, including NULL, prohibits this action.

VIEW_CONV

The VIEW_CONV table defines which groups of users can see which named conversions in the Telescope system. The table acts as a join table between the USERS table and the NAMED_CONV table. The presence of a record in this table indicates that the given group can see the given named conversion. The following chart describes the columns in the VIEW_CONV table.

Field	Data Type	Description
group_name	nchar(32)	The name of the user group for which the visibility applies. In Telescope 9.1 group_name is updated to nvarchar(32)

id	integer	The ID of the named conversion for which this visibility record applies.

VIEW_FIELDS

The VIEW_FIELDS table defines which groups of users can see and/or edit which fields in the EXTRA_COLUMNS table. The table acts as a join table between the USERS table and the EXTRA_COLUMNS table. The presence of a record in VIEW_FIELDS indicates that the user group can see the field. Other columns in this table determine whether the user group can edit the field, and whether the name of the field should be displayed differently for this particular group. The following chart describes the columns in the VIEW_FIELDS table.

Field	Data Type	Description
user_name	nvarchar(32)	Cross-reference to the user_name field in the USERS table, of the Group this refers to.
view_order	integer	Integer that represents the ordering of the field for this group or user's view.
edit_yn	nchar(1)	"Y" in this field indicates that the user or group can edit this field. Any other value means they cannot.
column_idx	short integer	Cross-reference to the ID field in the EXTRA_COLUMNS table.
nm_override	nchar(32)	If not empty, the value in this field is the name that will be used to display the field for the group. If it is empty, the field will use the default display name, as defined in EXTRA_COLUMNS.

VIEW_FM

The VIEW_FM table defines which groups of users can see which file migration policies in the Telescope system. The table acts as a join table between the USERS table and the FM_POLICIES table. The presence of a record in this table indicates that the given group can see the given file migration policy. The following chart describes the columns in the VIEW_FM table.

Field	Data Type	Description
group_name	nvarchar(32)	The name of a user group that can be assigned to a file migration policy.
fm_name	nchar(64)	The name of the file migration policy.

VIEW_FORMS

The VIEW_FORMS table defines which groups of users can see which form searches in the Telescope system. The table works as a join table between the USERS table and the FORM_SEARCH table. The presence of a record in this table indicates that the given group can see the given form search. The following chart describes the columns in the VIEW_FORMS table.

Field	Data Type	Description
user_name	nvarchar(32)	Cross-reference to the user_name field in the USERS table, of the Group this refers to.
search_id	integer	Cross-reference to search_id field in the FORM_SEARCH table.

VIEW_GROUPS

The VIEW_GROUPS table defines what other groups a given group can see, in the event that their visibility should be limited. The interpretation of the records in this table is subtly different from the other "VIEW_..." tables, in that the absence of any records for a particular group in this table indicates that the group has unlimited visibility (i.e., that members of the group can see all other groups). There may be several VIEW_GROUPS entries for each user group whose visibility is being limited. The entries in this table describe all the groups that a given group can see, including itself. For example, if a given group of users should only be able to see their own group, the VIEW_GROUPS table would contain a single entry for this group, where the GROUP_NAME and VISIBLE_GROUP fields are equal.

The following chart describes the columns in the VIEW_GROUPS table.

Field	Data Type	Description
group_name	nvarchar(32)	Cross-reference to the user_name field in the USERS table, of the Group this refers to.
visible_group	nvarchar(32)	User group that group_name can see. This is also a cross-reference to the user_name field in the USERS table.
id	integer	Unique ID generated by the system for each record in the table.

VIEW_HIER

The VIEW_HIER table defines which groups of users can see which hierarchical (tree) searches in the system. The table works similarly to the VIEW_FIELDS table, acting as a join table between the USERS table and the

HIERARCHIES table. The presence of a record in this table indicates that the given group can see the given hierarchy. The following chart describes the columns in the VIEW_HIER table.

Field	Data Type	Description
user_name	nvarchar(32)	Cross-reference to the user_name field in the USERS table, of the Group this refers to.
hier_id	integer	Cross-reference to the hier_id field in the HIERARCHIES table.

VIEW_METHODS

The VIEW_METHODS table stores the list of download methods available to each user. The following chart describes the columns in the VIEW_METHODS table.

Field	Data Type	Description
user_name	nvarchar(32)	The username. Cross-reference to USERS.user_name.
method_id	integer	The name of the download method available to this user. Cross-reference DL_METHODS table.

VIEW_REND

The VIEW_REND table is used to govern which user groups can see which renditions. Like the other tables of its type, this table is a simple joining table between the RENDITIONS and the USERS tables. The presence of a record in VIEW_REND indicates that the given group can see the given rendition. The following chart describes the columns in the VIEW_REND table.

Field	Data Type	Description
user_name	nvarchar(32)	Cross-reference to the user_name field in the USERS table. This user name is the name of a Group user.
rendition	integer	Cross-reference to the rend_id field in the RENDITIONS table.

VIEW_RM

(For Orchestration) The VIEW_RM table defines which groups can see which route maps in Telescope. The table acts as a join table between the USERS table and the WS_ROUTEMAPS table. The following chart describes the columns in the VIEW_RM table.

Field	Data Type	Description
user_name	nvarchar(32)	User group. Cross-reference to USERS.user_name.
rm_id	integer	The route map ID. Cross-reference to the ID column of the WS_ROUTEMAPS table.

VIEW_SOURCES

The VIEW_SOURCES table contains the view privileges that link user groups to sources in the Lookup Broker. A user group must have an entry in this table for a specific lookup field in order to be able to access the Lookup Broker functionality. Users in groups without an entry for a specific field will see a grayed out Lookup Broker icon that does not function beside the lookup field in question. For more information on defining lookups, see the *Telescope Administrator's Reference Manual*.

The following chart describes the columns in the VIEW_SOURCES table.

Field	Data Type	Description
vs_id	integer	A unique key for this table. This is an administrative ID, generated by the Telescope Admin application.
column_id	integer	The ID of the column for which the source applies, which is a cross reference to the ID column in the EXTRA_COLUMNS table.
group_name	nvarchar(32)	The user group for which this privilege applies.
source_name	nvarchar(32)	The name of the source obtained from the Lookup Broker, which the user group has permission to see.

VIEW_TRACKS

The VIEW_TRACKS table stores the list of text tracks available to each user group. The following chart describes the columns in the VIEW_TRACKS table.

Field	Data Type	Description

group_name	nvarchar(32)	The user group. Cross-reference to USERS.user_name.
track_id	big integer	ID of the text track. Cross-reference to the track_id column in the VL_TRACKS table.
edit_yn	nchar(1)	If this value is "Y" this user has edit permission for the track. Default is "N".

VIEW_VIDEOMGR

The VIEW_VIDEOMGR table stores the list of rights available to each user group for different functions of the Video Manager interface. Valid ENTITY entries are KEY FRAMES, CLIPS and PROXIES. An entry in this table signifies the user group has at least VIEW privileges to the functionality referenced in the ENTITY column. The following chart describes the columns in the VIEW_VIDEOMGR table.

Field	Data Type	Description
group_name	nvarchar(32)	The user group name. Cross-reference to USERS.user_name
entity	nchar(10)	This field determines how to view the attached collection (catalog). Possible values are: **keyframes** – sets the permission for the keyframes section. **Clips** – sets the permission for the clips section. **Proxies** – sets the permission for the proxies section
edit_yn	nchar(1)	If this value is "Y" this group has edit permissions for the functionality referenced by the value in the entity column. Any other value, including NULL, prohibits editing.

VIEW_VL_ANNOTATIONSETS

The VIEW_VL_ANNOTATIONSETS table stores the list of rights available to each user group for annotation sets. The following chart describes the columns in the VIEW_VL_ANNOTATIONSETS table.

Field	Data Type	Description
group_name	nvarchar(32)	User group. Cross-reference to USERS.user_name.
set_id	integer	The set ID. Cross-reference to the set_id column of the VL_ANNOTATIONSETS table.
edit_yn	nchar(1)	If this value is "Y" this group has edit permissions for the annotation set referenced by the value in the set_id column. Any other value, including NULL, prohibits editing. Default is "N".

VIEW_WELCOMEPAGES

The VIEW_WELCOMEPAGES table stores the list of Welcome Pages available to each user group. The following chart describes the columns in the VIEW_WELCOMEPAGES table.

Field	Data Type	Description
group_name	nvarchar(32)	Group name. Cross-reference to USERS.user_name.
page_id	integer	ID of the Welcome Page. Cross reference to the the page_id column of the WELCOME_PAGES table.

Collection Tables

Telescope collections (catalogs) are ad-hoc collections of assets that allow users to organize their work environment to suit their needs. Collections can be private or public (meaning they are visible to other users or groups).

Note: Collections were previously called "catalogs" in earlier releases of Telescope.

M_LB_ITEMS

The M_LB_ITEMS table contains the actual record (asset) contents of the collections. The following chart describes the columns in the M_LB_ITEMS table.

Field	Data Type	Description
lb_id	integer	ID of the collection in which this item belongs. Cross-reference to the ID column in the M_LIGHTBOXES table.
lb_order	integer	Position of the item in the collection (numbered from 1 .. n, where n is the number of items in the collection).
record_id	integer	The asset's record_id in the EDITORIAL table.

M_LIGHTBOXES

This table contains Telescope's Collections. A collection (formerly called "catalog") is made up of a collection of assets that are saved in the database. The M_LIGHTBOXES table together with the M_LB_ITEMS table defines a Telescope collection.

Some interesting customizations can be made with the M_LIGHTBOXES table. For example, it is possible to have events trigger the creation of a new collection. Imagine that there is an automated news feed linked into the Telescope environment. A scheduled job could be created (cron job or DBMS_JOB in Oracle) that runs early every morning (1 a.m.). This job would create a new public collection containing all the assets that were obtained through the news feed the day before.

The following chart describes the columns in the M_LIGHTBOXES table.

Field	Data Type	Description
id	integer	An identifier to uniquely identify the collection, generated by Telescope.
lb_name	nvarchar(256)	Collection name assigned by the user.

Field	Data Type	Description
owner	nvarchar(32)	The user who created the collection. This value will always be filled in, even if the collection is a public collection. (If the user is deleted from the USERS table, this value will remain.)
userid	integer	*DEPRECATED.*
public_yn	nchar(1)	One-character flag that contains the value "Y" if the collection is a public collection and available to other users, or "N" if the collection is private and only available to the owner.
password	nvarchar(20)	The password for the collection. An empty password (or NULL) means the collection is not password protected.
group_id	integer	*DEPRECATED.*
create_date	datetime (timestamp)	Date the collection was created. Note: The original system create and modify dates may not always be preserved when a file is imported, because of differences in data handling between different operating systems.
mod_date	datetime (timestamp)	Date the collection was last modified. Note: The original system create and modify dates may not always be preserved when a file is imported, because of differences in data handling between different operating systems.
parent_id	integer	The ID of the collection which contains this collection entry (the parent). A non-zero value here indicates this collection is 'nested'.
type	nchar(1)	Specifies whether or not the collection is a smart collection. If set to 'S', the collection is a smart collection. A NULL values means the collection is not a smart ca collection talog.
lb_xml	nvarchar(max)	If the collection is a smart collection, this value (represented in XML format) will have the command used to generate the search used to generate the collection contents.
solr_search	nvarchar(max)	If the collection is a smart collection and the Solr search method is implemented, this value contains the Solr query to generate the collection contents.
search_criteria	nvarchar(max) (NCLOB for Oracle)	Stores smart collection criteria.

Welcome Pages Tables

WELCOME_ICONIC_LEVELS

The WELCOME_ICONIC_LEVELS table contains information about each level of the Welcome Page iconic searches. The following chart describes the columns in the WELCOME_ICONIC_LEVELS table.

Field	Data Type	Description
search_id	integer	A reference to the search_id column in the WELCOME_ICONIC_SEARCHES table, of the iconic search to which this level belongs.
level_no	integer	An integer from 1 .. *n*, indicating the 'depth' of the level in the iconic search. This value will match the value of the level_id column in the HIER_LEVELS table for the level being defined.
display_type	nchar(4)	The desired display type for the level, which governs how elements are displayed to the user. There are 3 possible values for this field: **icon** – Show the graphical icon only for elements on this level. **Text** – Show the textual value of the elements for this level, without the graphical icon (this is still different than a traditional textual tree search, however, as the text items are displayed in a tabular format, rather than as a list with 'reveal' controls). **Both** – Show both the graphical icon and the text. In the absence of a valid value in this field, the default is 'both'.
level_title	nvarchar(255)	An optional display title for the level. If this column is empty or NULL, it will be used in the iconic search display as the title for the level when the user navigates to it during the search.
default_icon	binary	An optional default icon for the level. If not NULL, this column will contain a PNG image (of any size) that will be used as the icon for any elements in the level that do not have an explicit icon listed in the WELCOME_ICONS table. The PNG format is used in order to permit transparency for the icons.
default_icon_ description	nvarchar(4000)	An optional default icon description for the level. If not NULL, this value will be used as the descriptive text for any icon values selected on this level that do not explicitly specify their own descriptive text in the WELCOME_ICONS table.

default_action	nvarchar(10)	An optional default action to take when the user clicks on an icon element in this level. If this column is empty or NULL, the default action is assumed to be 'Reveal' if the level is not the last level in the tree. Otherwise, the default value is 'Search'. If not NULL, there are 4 valid values for this field: **Reveal** – When the user clicks on an icon element, reveal the next level of the search. **Search** – Execute the search indicated by the selected element. **Details** – Execute the search indicated by the selected element, and display the document info details page for the first asset found. **Preview** – Execute the search indicated by the selected element, and display the preview page for the first asset found.

WELCOME_ICONIC_SEARCHES

The WELCOME_ICONIC_SEARCHES table lists the iconic search information that can be used for display on a welcome page. Telescope administrator can define as many welcome page iconic searches as they like, and can reference them in the welcome page HTML file, using the drop-in 'iconic search' component. The following chart describes the columns in the WELCOME_ICONIC_SEARCHES table.

Field	Data Type	Description
search_id	integer	The unique key for this table – an integer value that identifies this iconic search. Telescope Admin creates this identifier when a new iconic search is created. In the 'iconic search' drop-in component on the welcome page HTML, this ID can be referenced using the search_id attribute in the WOD file, to link to this specific search.
search_name	nvarchar(64)	The displayed name for this search. This is the name shown to the administrator in Telescope Admin for the search, and additionally, in the 'iconic search' drop-in component on the welcome page HTML. This name can be referenced using the search_name attribute in the WOD file, to link to this specific search. Because of this requirement, this field will have a unique index defined on it in the database, and Telescope Admin will ensure that two iconic searches with the same name cannot be created.
div_width	integer	The width (in pixels) of the area to be reserved for the search. This will result in a DIV tag of the given width being inserted into the HTML.

div_height	integer	The height (in pixels) of the area to be reserved for the search. This will result in a DIV tag of the given height being inserted into the HTML.
icon_columns	integer	The number of columns that will be shown in the table of icons presented for each level in the search.
icon_rows	integer	The number of rows of icons to display per batch in the search. If the level display for any level of this search shows more icons than can be displayed in this number of rows, a batch navigator will be used to move through the level result in batches.
use_tree	integer	A reference to the hier_id column in the HIERARCHIES table, of the tree search which should be used as the basis for this iconic tree. The 'iconic search' drop-in component will communicate with the Tree Broker, requesting the tree specified by this value, to obtain the data to populate the various levels of the iconic search.

WELCOME_ICONS

The WELCOME_ICONS table contains all of the icons to display for a welcome page iconic search. The following chart describes the columns in the WELCOME_ICONICS table.

Field	Data Type	Description
search_id	integer	A reference to the search_id column in the WELCOME_ICONIC_SEARCHES table, of the iconic search to which icon belongs.
level_no	integer	An integer from 1 .. *n*, indicating the 'depth' of the level in the iconic search. This value will match the value of the level_id column in the HIER_LEVELS table for the level being defined.
valuestr	nvarchar(255)	The icon element's value, which will be matched against the values returned by the Tree Broker for the level. Since this matching is done case insensitively, the capitalization of this value is unimportant.
icon_image	binary	The icon image. This is stored in PNG format (to allow for transparency), of any size.
icon_description	nvarchar(4000)	An optional description for the icon element. If not NULL, this value will be used as the descriptive text for the element, when the user selects it in the search.

action_type	nvarchar(10)	The desired action to take when the user clicks on this element. If this column is empty, NULL, or is not one of the listed values, it is assumed to contain 'default'. There are 6 possible values for this field: **Default** – Perform the default action for the level, as defined by the *default_action* column in WELCOME_ICONIC_LEVELS. **Reveal** – Reveal the next level of the search. **Search** – Execute the search indicated by the element. **Catalog** – Display a collection (catalog) (see the action_id column for information). **Details** – Display a document info details page (see the action_id column for information). **Preview** – Display a preview page (see the action_id column for information).
action_id	integer	An identifier that defines, in conjunction with the action_type column above, what should happen when the user clicks on the element. For an action_type of 'catalog', this value is the ID of a collection (catalog) that should be displayed to the user. If action_id references a collection that doesn't exist, or that the logged-in user cannot see, an error is displayed to the user. For an action_type of 'details' or 'preview', this value is the record_id for the asset whose information should be shown. If this value references an asset that doesn't exist, or that the logged-in user cannot see, an error is displayed to the user. However, if this value is NULL, then the element's search is executed, and the appropriate page is shown for the first asset found.

WELCOME_PAGES

The WELCOME_PAGES table stores the list of Welcome Pages available in the system. The following chart describes the columns in the WELCOME_PAGES table.

Field	Data Type	Description
page_id	integer	The ID for the Welcome Page.
page_name	nvarchar(64)	The convenience name of the Welcome Page.
html_file	nvarchar(255)	The file name of the Welcome Page. It must reside in the …\tsweb.woa\Contents\Resources\WelcomePages folder.
lang_id	nchar (10)	Identifies the language (from the language_local table, lang_id column). For example, en_US.

Messaging Tables

Telescope's messaging system works by storing 'messages' in the database, and having each logged-in user periodically poll the database to see if they have new messages. Attaching an asset to a message is simply done by passing a reference to the attached asset's RECORD_ID value, rather than sending the potentially large asset through the network; as would happen with standard SMTP messaging.

M_ACTIONS

The M_ACTIONS table stores the list of message actions available on the system, along with the code that is actually executed when that message action is selected. The following chart describes the columns in the M_ACTIONS table.

Field	Data Type	Description
action_code	nvarchar(64)	The name used to refer to the action within Telescope.
display_name	nvarchar(255)	The displayed name of the action. This will become the button label within a message.
action_script	nvarchar(4000)	The SQL script to be executed at runtime when the user clicks on the Action button within a message. Replacement parameters available to be passed in the script are: • MSG_ID – Message ID from the M_MESSAGES table • UN – user_name of the user from the USERS table • UG – user_name (group name) of the user from the USERS table
system_action	nchar(1)	If the value is "Y" the action is one of the system supplied actions. In that case, any action script in this record will be ignored and replaces by the built-in functions.
service_action	nchar(1)	Indicates if the action is for Orchestration Services: **N** Regular message action (Default value). **Y** Service action.
decision_id	integer	The decision ID from the WS_SERVICEDECISIONS table.
comment_label	nvarchar(255)	The comment label to display requesting input when the user takes an action. A "NULL" value means user input is not required.
service_id	integer	This is used for Orchestration. If the message is tied to an Orchestration service, the service_id identifies what running workflow the message is tied to.

Field	Data Type	Description
require_assets_YN	nchar(1)	This is used for Orchestration. The option in the route map (within Telescope Admin) for "Require Assets" is captured in the current service within require_assets_YN. This means that the user is not supposed to be allowed to proceed with the flow without picking one or more assets to continue with. Default is "N".

M_ATTACHMENTS

The M_ATTACHMENTS table stores the relationship between messages and their attached documents. For each document attached to a particular message, there will be an M_ATTACHMENTS record, which links the document and the message. The following chart describes the columns in the M_ATTACHMENTS table.

Field	Data Type	Description
msg_id	integer	Cross-reference to the ID column of the M_MESSAGES table.
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.

M_MESSAGES

The M_MESSAGES table contains the information about each Telescope message in the system. Messages can be sent from one user to another, and the contents and information about the messages are stored in six tables: M_MESSAGES, M_MSGTEXT, M_ATTACHMENTS, M_RECIPIENTS, M_MSGACTIONS, and M_ACTIONS.

Note: Messages sent using the Telescope internal messaging system are not tied to email. The User Preferences can be configured to send an SMTP email alerting the Telescope message recipient that a message waiting.

Environments where some Telescope users may only make use of the system periodically may have a need to alert those users when a message is waiting for them.

The following chart describes the columns in the M_MESSAGES table.

Field	Data Type	Description
id	integer	Unique identifier generated by the system for each message.
from_user	nvarchar(32)	User name of the user who sent the message.

Field	Data Type	Description
userid	integer	*DEPRECATED.*
to_user	nvarchar(32)	*DEPRECATED.*
to_userid	integer	*DEPRECATED.*
timesaved	datetime (timestamp)	Date and time the message was saved (sent).
read_flag	nchar(1)	*DEPRECATED.*
headline	nvarchar(255)	Short description or the subject line for the message.
template_id	integer	ID of template contained in the M_TEMPLATE table.

M_MSGACTIONS

The M_MSGACTIONS table stores the relationship between messages and any message actions available to the user upon viewing the message (for example, approve or deny). The following chart describes the columns in the M_MSGACTIONS table.

Field	Data Type	Description
msg_id	integer	Cross-reference to the M_MESSAGES table.
action_code	nvarchar(64)	Cross-reference to the M_ACTIONS table.

M_MSGTEXT

The M_MSGTEXT table contains the actual text of each message. To allow for unlimited-length messages, each message in the M_MESSAGES table can have as many MSGTEXT entries as required to hold the entire message. When retrieved in order, the MSGTEXT records contain the entire text of the message. The following chart describes the columns in the M_MSGTEXT table.

Field	Data Type	Description
msg_id	integer	Cross-reference to the M_MESSAGES table.

Field	Data Type	Description
txt_id	integer	Ordering of the msgtext records in the message. For example, a sufficiently long message might contain three msgtext entries, all with the same msg_id field, and with txt_id fields containing the values 1, 2, and 3, in order. Retrieving the msgtext entries ordered by txt_id will retrieve the message, in the correct sequence, in (approximately) 2k chunks. When saving a message, the message should be divided up into (approximately) 2k chunks, and saved in order, setting txt_id appropriately for each chunk saved.
msgtext	nvarchar(max)	Actual text of the message.
lang_id	nchar(10)	Identifies the language (from the language_local table, lang_id column). For example, en_US.

M_RECIPIENTS

The M_RECIPIENTS table stores the relationship between messages and their recipients. The following chart describes the columns in the M_RECIPIENTS table.

Field	Data Type	Description
msg_id	integer	Cross-reference to the M_MESSAGES table.
recipient_id	integer	An ordinal reference to the message recipients and the unique ID for this table.
user_name	nvarchar(32)	The user name of the recipient.
read_date	datetime (timestamp)	Once the user opens the message this date time stamp is set.
todo_flag	nchar(1)	A "Y" indicates the message has a Message Action attached. See action_taken below. A "D" indicates the message was deleted from the inbox by the recipient.
action_taken	nvarchar(64)	The action code corresponding to the Action the user has clicked in the message. Cross-reference to the M_ACTIONS table.
due_date	datetime (timestamp)	The due date of the current message if the message is part of the Orchestration Services.

M_TEMPLATE

The M_TEMPLATE table stores localizable templates for system-generated messages.

The following chart describes the columns in the M_TEMPLATE table.

Field	Data Type	Description
id	integer	Cross-reference to the M_MESSAGES table, template_ID column.
template_name	nchar(48)	Name of the specific template being described. Each template setting is a keyword/value pair, so the keyword column is unique for a particular user or group.
template_text	nvarchar(max)	The preference setting itself. The contents of this column depend on the keyword associated with it.
template_type	nchar(4)	The origin of the template. For example, FFPL=FlipFactory Player (Deprecated) IB= Ingest Broker MB=Message Broker. TRPL=Transformation Platform
lang_id	nchar(10)	Identifies the language (from the language_local table, lang_id column). For example, en_US.

The following are some examples of default template text used by Telescope (for lang_id=en_US):

template_type	template_name	template_text
MB	emailNotificationBody	'<!MESSAGE_TEXT!>' + CHAR(10) + CHAR(10) + 'Follow the link below to log into Telescope and see your messages:' + CHAR(10) + '<!LOGIN_URL!>'
MB	emailOriginalSenderFooter	CHAR(10) + CHAR(10) + 'The original sender of this email is: <!ORIGINAL_SENDER!>, please reply to this account.' + CHAR(10)
FFPL	flipSuccessSubject	"Video asset completed processing"
FFPL	flipErrorSubject	"Video asset processed, WITH ERRORS"
FFPL	flipSuccessBody	"The attached video asset was processed without errors by the Flip Factory server."

Orchestration Services Tables

These tables support the Telescope Orchestration Services. For details on how to set up and use orchestration, see the chapter in the *Telescope Administrator's Reference Guide*.

WS_ARCHIVE

The WS_ARCHIVE table stores information about completed services. The following chart describes the columns in the WS_ARCHIVE table.

Field	Data Type	Description
id	integer	A unique key for this table.
from_category	nvarchar(64)	The name of the route map category the service is based on from the rm_name field from the WS_ROUTEMAPS table.
archive_date	datetime (timestamp)	The date and time the service was moved from the service trace tables and added to the archive.
service_text	nvarchar(max)	XML formatted text that contains all of the information about the service at the time it was moved to the archive.

WS_DECISIONS

The WS_DECISIONS table defines decisions for each junction. The following chart describes the columns in the WS_DECISIONS table.

Field	Data Type	Description
id	integer	A unique identifier for the decision.
junction_id	integer	The ID of the junction this decision belongs to from the WS_JUNCTIONS table.
name	nvarchar(64)	The name of the decision.
threshold	integer	The threshold percentage of users who must choose this decision to activate it.
comment_label	nvarchar(255)	The comment label to display requesting input when the user makes the decision. "NULL" means the user input is not required.
side_effect	nvarchar(max)	The XML definition of the side effect of this decision, if any.

linked_to	integer	The ID of the junction this decision points to. "NULL" means "End of Route".
require_assets_YN	nchar(1)	This indicates if the user must select one or more assets before clicking on one of the decision buttons for this junction of the workflow. If set to Yes, the system will not allow the flow to proceed if the user does NOT pick an asset and clicks a decision button. Default is "N".

WS_J_NOTIFICATIONS

The WS_J_NOTIFICATIONS table defines the junction users who should be notified if a service is delayed (or a milestone is missed) at a given junction. The following chart describes the columns in the WS_J_ NOTIFICATIONS table.

Field	Data Type	Description
junction_id	integer	The ID of the junction from the WS_JUNCTIONS table.
user_name	nvarchar(32)	The user who should be notified.
notification_type	integer	Defines the type of notification to send. Valid values are: **1** Notify the user if the service is delayed **2** Notify the user if a milestone is missed

WS_J_USERS

The WS_J_USERS table defines users who can move a service forward at a decision point. The following chart describes the columns in the WS_J_USERS table.

Field	Data Type	Description
junction_id	integer	The ID of the junction from the WS_JUNCTIONS table.
user_name	nvarchar(32)	The user responsible for the decision.
required	nchar(1)	Indicates whether this user must provide a decision at this junction. Valid values are: **Y** A decision is required **N** A decision is optional (Default value)

WS_JUNCTIONS

The WS_JUNCTIONS table defines the sequences of steps that assets will follow through a given routing. The following chart describes the columns in the WS_JUNCTIONS table.

Field	Data Type	Description
id	integer	A unique identifier, generated automatically, for the junction.
rm_id	integer	The route map ID to which this junction belongs.
name	nvarchar(64)	The name of the junction.
description	nvarchar(4000)	A description of the junction.
time_alocated	integer	The amount of time allocated to this junction.
time_unit	integer	An integer representing the unit of time. Valid values are: **1** Hours **2** Days **3** Weeks **4** Months
normal_compression	nchar(1)	Indicates is the timeline acceleration is set to normal or compressed. **Y** Compressed (Default value) **N** Normal
milestone	integer	Indicates whether the junction may have a milestone date associated with it. **1** Never **2** Optional **3** Always
delay_notifications	nchar(1)	Indicates whether the service owner should be notified if the service is delayed at this junction. If "Y", notify the owner; default "N".
missing_notification	nchar(1)	Indicates whether the service owner should be notified if the milestone is missed at this junction. If "Y", notify the owner; default "N".

WS_RM_NOTIFICATIONS

The WS_RM_NOTIFICATIONS table defines users who should be notified when a service based on this route map fail to execute. The following chart describes the columns in the WS_RM_NOTIFICATIONS table.

Field	Data Type	Description
rm_id	integer	The route map ID from the WS_ROUTEMAPS table.
user_name	nvarchar(32)	The user to be notified.
notification_type	integer	The type of notification. Valid values for this field are: **1** Service Monitor **2** Notification user

WS_ROUTEMAPS

The WS_ROUTEMAPS table defines the sequences of steps that assets will follow through a given routing. The following chart describes the columns in the WS_ROUTEMAPS table.

Field	Data Type	Description
id	integer	A unique identifier for the route map.
type	nvarchar(64)	The route map type.
name	nvarchar(64)	The name of the route map (the combination of rm_type and rm_name must be unique).
description	nvarchar(4000)	A free-form description of the route map.
notifyowner	nchar(1)	Indicates whether the service owner should also be notified of service execution failures. Valid values are "Y" and "N"; the default value is "N".
owner	nvarchar(32)	The user who created the route.

WS_S_NOTIFICATIONS

The WS_S_NOTIFICATIONS table defines the users who can monitor a running service. The following chart describes the columns in the WS_S_NOTIFICATIONS table.

Field	Data Type	Description
service_id	integer	The ID of the service from the WS_SERVICES table.
user_name	nvarchar(32)	The user who can monitor the service.
notification_type	integer	Defines the type of notification to send. Valid values are: **1** Notify the user if the service is delayed **2** Notify the user if a milestone is missed

WS_SERVICEASSETS

The WS_SERVICEASSETS table stores the assets attached to the running service. The following chart describes the columns in the WS_SERVICEASSETS table.

Field	Data Type	Description
service_id	integer	The IDof the service from the WS_SERVICES table.
record_id	integer	The record_id of the asset from EDITORIAL table.

WS_SERVICEDECISIONS

The WS_SERVICEDECISIONS table stores information about the decisions for a currently running service. The following chart describes the columns in the WS_SERVICEDECISIONS table.

Field	Data Type	Description
id	integer	A unique identifier for the decision.
junction_id	integer	The ID of the junction this decision belongs to from the WS_SERVICEJUNCTIONS table.
name	nvarchar(64)	The name of the decision.
threshold	integer	The threshold percentage of users who must choose this decision to activate it.
comment_label	nvarchar(255)	The comment label to display requesting input when the user makes the decision. "NULL" means the user input is not required.
side_effect	nvarchar(max)	The XML definition of the side effect of this decision, if any.
linked_to	integer	The ID of the junction this decision points to. "NULL" means "End of Route".
require_assets_YN	nchar(1)	Similar to the WS_DECISIONS table.require_assets_YN. The WS_DECISIONS table records the settings for the route map configuration. When a service is kicked off, the settings are copied from the WS_DECISIONS table into the WS_SERVICEDECISIONS table for use for the running flow.

WS_SERVICEJUNCTIONS

The WS_SERVICEJUNCTIONS table stores information about the junctions for a given service. The entries are entered when each service starts and removed when each service is completed (or stopped). The following chart describes the columns in the WS_SERVICEJUNCTIONS table.

Field	Data Type	Description
id	integer	A unique identifier, generated automatically, for the junction.
service_id	integer	The service ID to which this junction belongs.
name	nvarchar(64)	The name of the junction.
description	nvarchar(4000)	A description of the junction.
time_alocated	integer	The amount of time allocated to this junction.
time_unit	integer	An integer representing the unit of time. Valid values are: **1** Hours **2** Days **3** Weeks **4** Months
normal_compression	nchar(1)	Indicates is the timeline acceleration is set to normal or compressed. **Y** Compressed (Default value) **N** Normal
milestone	integer	Indicates whether the junction may have a milestone date associated with it. **1** Never **2** Optional **3** Always
delay_notifications	nchar(1)	Indicates whether the service owner should be notified if the service is delayed at this junction. If "Y", notify the owner; default "N".
missing_notification	nchar(1)	Indicates whether the service owner should be notified if the milestone is missed at this junction. If "Y", notify the owner; default "N".

WS_SERVICES

The WS_SERVICES table stores the information for each service. The following chart describes the columns in the WS_SERVICES table.

Field	Data Type	Description
id	integer	A unique identifier for the service.
type	nvarchar(64)	The route map type this service is based on.
name	nvarchar(256)	The name of the service.
description	nvarchar(4000)	The route map description from the WS_ROUTEMAPS table.
notifyowner	nchar(1)	The notification setting from the WS_ROUTEMAPS table.
msg_id	integer	The Telescope message identifier that represents the service.
owner	nvarchar(32)	The user who initiated the service.
start_date	datetime (MSSQL) or date (Oracle)	The date and time the service was initiated. This is the service "baseline" date for all the subsequent calculations to determine whether the service is running on schedule.
end_date	datetime (MSSQL) or date (Oracle)	The date and time the service completed; the value is NULL during the service's run.
tz_offset	integer	The TimeZone offset from UTC, in milliseconds. For example, New York standard time at -5 hours would be 5 x 60,000 = - 300,000 ms.

WS_SERVICETRACE

The WS_SERVICETRACE table stores trace information about the junctions for a running service. The following chart describes the columns in the WS_SERVICETRACE table.

Field	Data Type	Description
id	integer	A unique identifier for the trace.
junction_id	integer	The ID of the junction this decision belongs to from the WS_SERVICEJUNCTIONS table.
sequence_no	integer	The current junction's location in the running service.

type	integer	The type of junction trace describes. Valid values are: **1** Baseline **2** Real time
state	integer	The state of the junction. Valid values are: **1** Passed **2** Current **3** Future
start_date	datetime (MSSQL) or date (Oracle)	The time the junction was entered. For future junctions, the value will be the scheduled start date.
end_date	datetime (MSSQL) or date (Oracle)	The time the service left the junction. For current and future junctions, the end_date will be the scheduled due date.
milestone	datetime (MSSQL) or date (Oracle)	The milestone setting for this junction.
manual_advance	nchar(1)	Set to "Y" if the service owner or administrator manually advanced the service, otherwise the value defaults to "N".
decision_taken	integer	The decision ID from the WS_SERVICEDECISIONS table representing the decision made at this junction. This field is "NULL" for current and future junctions.

WS_SJ_NOTIFICATIONS

The WS_SJ_NOTIFICATIONS table defines the users who should be notified if the service is delayed (or a milestone is missed) at a given junction. The following chart describes the columns in the WS_SJ_NOTIFICATIONS table.

Field	Data Type	Description
junction_id	integer	The ID of the junction from the WS_SERVICEJUNCTIONS table.
user_name	nvarchar(32)	The user who should be notified.
notification_type	integer	Defines the type of notification to send. Valid values are: **1** Notify the user if the service is delayed **2** Notify the user if a milestone is missed

WS_SJ_USERS

The WS_SJ_USERS table defines users who can move a running service forward at a given decision point. The following chart describes the columns in the WS_SJ_USERS table.

Field	Data Type	Description
junction_id	integer	The ID of the junction from the WS_SERVICEJUNCTIONS table.
user_name	nvarchar(32)	The user responsible for the decision.
required	nchar(1)	Indicates whether this user must provide a decision at this junction. Valid values are: **Y** A decision is required **N** A decision is optional (Default value)

WS_ST_USERS

The WS_ST_USERS table stores the user decision information of past and current junctions. The following chart describes the columns in the WS_ST_USERS table.

Field	Data Type	Description
trace_id	integer	The service trace ID from the WS_SERVICETRACE table.
recipient_id	integer	The ID of the decision user.
user_name	nvarchar(32)	The user_name from users table.
required	nchar(1)	Indicates whether a decision is required. Value values are: **1** Required **2** Optional
read_date	datetime (MSSQL) or date (Oracle)	The date and time the user made the decision.
todo_flag	nchar(1)	A "Y" indicates the message has a message action attached. Default is "N".
action_taken	nvarchar(64)	The action_code from M_ACTIONS table.
comments	nvarchar(256)	The user's input comments.

Search Tables

(For details on setting up searching, see the *Telescope Administrator's Reference Manual*.)

Telescope offers several different types of searches:

KEYWORD ("SIMPLE") SEARCH: is an easy-to-use search that permits word-based searching on several database fields simultaneously.

ADVANCED SEARCH: is a 'power' search that permits the definition by the user of arbitrarily complex Boolean-logic based searches.

TREE (or HIERARCHICAL) SEARCH: is a very intuitive, 'drill-down' search, where the administrator defines a hierarchy of metadata fields that the user can explore in a tree-like display that makes browsing the contents of the database very simple.

FORM SEARCH is a named search set up by the administrator to provide a quick method for users to find what they are looking for.

CONTENT SEARCH: is a full-text search and retrieval tool that can query the text contents of many types of documents (word processing documents, video closed-caption data, etc.) and provide relevance-ranked results to the user.

The following tables are used to manage the various search capabilities of Telescope.

FORM_SEARCH

The FORM_SEARCH table stores information about the form searches defined by the administrator. Each search has an entry in the FORM_SEARCH table, and the fields that define the search form are in the FORM_SEARCH_FIELDS table. The following chart describes the columns in the FORM_SEARCH table.

Field	Data Type	Description
search_id	integer	Unique identifier for the specified search. This is an administrative ID, generated by the Telescope Admin application.
search_name	nvarchar(64)	Name of the search as shown to the users.

FORM_SEARCH_FIELDS

The FORM_SEARCH_FIELDS table stores information about the search forms defined by the administrator. Each search has an entry in the FORM_SEARCH table, and the fields that define the search form are in the FORM_SEARCH_FIELDS table. The following chart describes the columns in the FORM_SEARCH_FIELDS table.

Field	Data Type	Description
field_id	integer	Unique identifier for each row in the FORM_SEARCH_FIELDS table.
search_id	integer	Cross-reference to the FORM_SEARCH table. Indicates the search to which this field belongs.
field_order	short integer	Ordering on the search form of this field. This field contains a number from 1 .. n, where n is the number of fields in the form search.
col_name	nvarchar(64)	Name of the column (from the column_name field in the EXTRA_COLUMNS table) in the metadata model that represents the field being searched by this form search.
input_type	short integer	Type of input this field requires from the user. Valid values for this column are: **1** text input **2** popup menu **3** live popup menu
col_idx	integer	Cross-reference to the ID field in the EXTRA_COLUMNS table for the column with which the popup menu is associated.
init_value	nvarchar(255)	A default initial value.
open_paren	nchar(1)	A value of "Y" means that this row is a "(" character.
close_paren	nchar(1)	A value of "Y" means that this row is a ")" character.

Field	Data Type	Description
operator	integer	List of operators: **1** Is **2** Is Not **3** Less Than **4** Greater Than **5** Less Than Or Equal To **6** Greater Than Or Equal To **7** Contains **8** Does Not Contain **9** Starts With **10** Does Not Start With **11** Ends With **12** Does Not End With **13** User Choose
conjunction	integer	**1** AND **2** OR

FORM_SEARCH_VALUES

The FORM_SEARCH_VALUES table presents popup menus for any field in a form search which are of the "live popup" input type. The following chart describes the columns in the FORM_SEARCH_VALUES table.

Field	Data Type	Description
value_id	integer	A unique identifier for each row in the FORM_SEARCH _VALUES table.
field_id	integer	A reference to the field, in the FORM_SEARCH_FIELDS table, for which this value applies.
valuestr	nvarchar(255)	The actual value of the item in the popup menu. Items displayed in the search form's popup menus are ordered alphabetically on this column.

HIER_ITEMS (Deprecated)

DEPRECATED. The HIER_ITEMS table has been replaced by the HIER_LEVELS table described below.

HIER_LEVELS (Tree Search)

The HIER_LEVELS table contains information about each level of the hierarchy in tree searches. The following chart describes the columns in the HIER_LEVELS table.

Field	Data Type	Description
hier_id	integer	Cross reference to the HIERARCHIES table.
level_id	integer	The level in the hierarchy for which this record applies. This is a value from 1 .. *n* where *n* is the number of levels in the hierarchy.
level_definition	nvarchar(4000)	Contains the XML text of the level definition.

HIERARCHIES

The HIERARCHIES table describes the hierarchies of fields that have been set up by the administrator for tree searches in TSAdmin.

The following chart describes the columns in the HIERARCHIES table.

Field	Data Type	Description
hier_id	integer	Unique identifier for the hierarchy, generated automatically.
name	nchar(64)	Name describing the hierarchy to the users. This name can be anything the administrator wishes, but will default to the names of the fields used in the hierarchy, separated by colons, for example, "country:state:city".

SAVED_SEARCHES

The SAVED_SEARCHES table contains information about the global saved searches that have been added from either the Field Search or Keyword Search dialogs. The following chart describes the columns in the SAVED_SEARCHES table.

Field	Data Type	Description
id	integer	A unique key for this table, generated automatically.
search_name	nchar(32)	Name of the saved search as displayed to the user.

Field	Data Type	Description
search_type	nchar(1)	Flag that indicates which dialog the search was saved from, possible values are: **K** – Keyword search. **F** – Field Search dialog.
search_text	nvarchar(max)	Text of the search, the format of which is dependent on whether search_type is "F" or "K".
user_name	nvarchar(32)	The user creating the saved search.

SEARCH_INDEX_ACTIONS

The SEARCH_INDEX_ACTIONS table is used in the Solr Search method by the Indexing Broker to track indexing actions. Every time the Indexing Broker picks up a batch of record IDs to process, before starting the indexing process it writes a row for each record ID into the search_index_actions table.

The following chart describes the columns in the SEARCH_INDEX_ACTIONS table.

Field	Data Type	Description
record_id	integer	The Telescope asset record ID.

Field	Data Type	Description
status	smallint	An integer indicating the processing status of that record ID. When a record ID row is initially created, its status value is set to "-1". After data for a record ID is sent successfully to the Solr Multicore, the status is updated to value "0". Indexing failures produce negative values, as listed below. All record IDs with negative status numbers are periodically resent to Solr for indexing. Status Values: 0 Record ID was successfully sent to Solr for search indexing. -1 Indexing is pending. The record has been added to the queue for processing. -2 The record has to be reindexed because data in the Telescope database has changed. -3 The indexer could not connect to Solr. (Solr could be down.) -4 The Telescope database query timed out (it took over 30 secs). -5 There was a Telescope database error when retrieving data. -6 There is insufficient memory to hold the data retrieved from Telescope. **Note:** If a Child Indexing Broker fails while in the middle of processing a batch of assets, those assets may be left in an abnormal state where they will no longer be processed, even with a -1 status. In this case, these assets require manual intervention by an administrator to adjust the queue and reset from pending (-1) to a ready for processing (-2) status using the following SQL command: update search_index_actions set status = -2 where status = -1;
index_time	datetime	The time when the latest action was executed on the record ID.
user_name	nvarchar(32)	The user who initiated the action. (Typically, "System," or the Indexing Broker.)
broker_name	nvarchar(32)	The IP address of the machine running the Indexing Broker or Child Indexing Broker.

Field	Data Type	Description
cm_process_hash	nvarchar(64)	A unique value autogenerated to identify the particular change multiple process. The name of this column stands for "change multiple hash code" and is built based on the following values: [hash code of the current object]_[user session id], which makes it a unique value per user session, and the per change multiple process (to distinguish between the change multiple processes if there are more than one within the same session).

SEARCH_INDEX_LOG

The SEARCH_INDEX_LOG table is used for logging the information from the search_index_actions table. You do not need to refer to this table; it is used internally by the Solr Search method and should not be altered.

The following chart describes the columns in the SEARCH_INDEX_LOG table.

Field	Data Type	Description
record_id	integer	See the description for the SEARCH_INDEX_ACTIONS table.
status	smallint	See the description for the SEARCH_INDEX_ACTIONS table.
log_status	smallint	The status value for the transaction processed that had the entry added to this log table.
index_time	datetime	See the description for the SEARCH_INDEX_ACTIONS table.
log_time	datetime	The time when the entry was added to this log table.
user_name	nvarchar(32)	See the description for the SEARCH_INDEX_ACTIONS table.
log_user	nvarchar(32)	The user who completed the transaction and caused it to be logged. (This may be a system user.)
broker_name	nvarchar(32)	See the description for the SEARCH_INDEX_ACTIONS table.

MIMiX (Synchronization Broker) Tables

MMX_SYNC

The MMX_SYNC table keeps assets synchronized between the different databases in a Synchronization Broker implementation. This table reflects data from the system that made the most recent change.

The following chart describes the columns in the MMX_SYNC table.

Field	Data Type	Description
record_id	integer	Cross-reference to the asset's record_id in the EDITORIAL table.
orig_id	integer	The record_id of the asset in the originating database.
orig_source	nchar(32)	The name of the source database as defined in the PushSettings.plist file.

Distribution Broker Tables

DISTB_AUDIT_TRAIL

The DISTB_AUDIT_TRAIL table keeps a record of all distributed assets when the "Enable Audit Trail" option is checked on the contract definition page. The following chart describes the columns in the DISTB_AUDIT_TRAIL table.

Field	Data Type	Description
id	integer	Unique ID generated automatically on insert into the table. On Oracle, this is a plain integer column with an insert trigger to populate it from an Oracle sequence.
record_id	integer	Cross-reference to the asset's record ID in the EDITORIAL table.
rend_id	integer	The rendition ID of the distributed asset.
execution_time	datetime (timestamp)	The time and stamp when the asset was distributed.
connection_name	nvarchar(255)	The connection name used in the contract definition page.
contract_name	nvarchar(255)	The name of the contract.
dipiece_name	nvarchar(255)	The name of the Distribution Broker Destination I-Piece used to distribute the asset.

Field	Data Type	Description
dest_path	nvarchar(1000)	A URL that points to the distributed file's real destination location. For failure distributions, the dest_path will be empty. If the destination is a File Broker share, dest_path will be in the format: • filebroker:FBName/FBShareName/[NTFS]/dir1/dir2/filename If the destination is a local or network mounted drive, dest_path will be in the format (note that the Directory Path setting in the SendToLocal Destination I-Piece will control which form to use): • file:C:\dir1\dir2\filename • file://unc/dir1/dir2/filename If the destination is an FTP location, dest_path will be in the format: • ftp://domain.com:port/dir1/dir2/filename
conversion_string	nvarchar(3000)	The conversion string applied to the asset being distributed (if any).
message	nvarchar(3000)	A message indicating if the asset was successfully distributed. In case of an error this value will contain a detailed error message.

DISTB_DATA_RECOVERY

The DISTB_DATA_RECOVERY table grants the Distribution Broker the opportunity to distribute contracts based on events which happened when the broker is not running. This table stores events in the database which the Distribution will consume either immediately or, if it was down when the event happened, at the time it starts up.

Note that, there is a registry setting that enables and disables Data Recovery. It is available for both the Distribution Broker and the Connection Broker (DATA_RECOVERY_ENABLED). When either is set to 0 (default), data recovery will be disabled. When set to 1 for both the Distribution Broker and the Connection Broker, data recovery will be enabled.

The following chart describes the columns in the DISTB_DATA_RECOVERY table.

Field	Data Type	Description

Field	Data Type	Description
id	identity	Unique ID generated automatically on insert into the table. On Oracle, this is a plain integer column with an insert trigger to populate it from an Oracle sequence.
insertion_time	datetime (timestamp)	The time and stamp when the asset was distributed (or the contract was executed).
connection_name	nvarchar(255)	The connection name used in the contract definition page.
user_name	nvarchar(255)	The username initiating the contract.
action_code	integer	The type of action which was performed to execute the contract. Ingest, Metadata Changed, etc
recordrendition_ids	varchar(max)	A list of record ID / rendition ID pairs which is the list of assets onto which the contract will operate. The format is: • <rec_id>:<rend_id> <rec_id>:<rend_id> <rec_id>:<rend_id>… For example: 123:1 3452:2 9985:1

Miscellaneous Brokers

Interoperability Broker Tables

INTEROP_EVENT_QUEUE

This table is used for functional rules to notify the Connection Broker of changes, so that the Connection Broker can relay these events to other listeners.

Field	Data Type	Description
event_id	integer	Unique ID generated automatically on insert into the table.
queue_date	datetime (timestamp)	The date and time when the event was added.
client_subscription	integer	The client subscription ID passed by the functional rule.

Other Interoperability Tables (For Future Use)

The following Interoperability Broker tables are intended for future use, and are not being used at this time:

- IOB_AUTHORIZATION
- IOB_CLIENT_SUBSCRIPTION_ASSETS
- IOB_DELETE_LOG
- IOB_DELETE_QUEUE
- IOB_EDITORIAL
- IOB_IOSYS_INGEST_LOG
- IOB_IOSYS_INGEST_QUEUE
- IOB_PARTIAL_RESTORE_LOG
- IOB_PARTIAL_RESTORE_QUEUE
- IOB_RESTORE_LOG
- IOB_RESTORE_QUEUE

Rest Broker Tables (For Future Use)

The following Rest Broker tables are intended for future use, and are not being used at this time:

- EV_CLIENT
- EV_CLIENT_AUTH
- EV_CLIENT_SUBSCRIPTION
- EV_EVENTS_LOG
- EV_SUBSCRIPTION
- EV_SUBSCRIPTION_ACTION

Queue Broker and Connection Broker Tables (For Future Use)

The following Queue Broker and Connectivity Broker tables are intended for future use, and are not being used at this time:

- CB_ACTION_CODES
- CB_EVENT_QUEUE
- CB_EVENTS
- QB_PROCESSES
- QB_QUEUE

System Tables

Telescope's system tables serve diverse needs within the Telescope system, permitting Telescope to manage its internal structure, storing user and database preferences, managing File Migration Policies, metadata templates, named conversions, etc. The following tables are used for Telescope system maintenance.

AUDIT

The AUDIT table is used to track administrative changes made by users. This table keeps track of users making the following actions:

- Creating and updating new users and groups (deleting is not tracking)
- Creating, updating, and deleting renditions
- Deleting new fields
- All other inserting, updating, and deleting actions in:
    - Announcements
    - Cross-platform
    - Download Methods
    - File Migration Policies
    - Fulfillers
    - Message Actions
    - Named Conversions
    - Functional Rules
    - Searches
    - Video Manager
    - Watermarks
    - Welcome Pages

The following chart describes the columns in the AUDIT table.

Field	Data Type	Description
AuditID	Identity (integer)	Unique ID generated automatically on insert into the table.

Field	Data Type	Description
UpdateDate	Datetime	The date/time stamp when the action took place.
DBUser	nvarchar(128)	The database account used to make the changes, (TSAdmin uses a single account for all operations)
TelescopeUser	nvarchar(128)	The Telescope account (user name) making the change.
Type	char(1)	Can be one of the following: **U** – Update action. **I** – Insert action. **D** – Delete action.
TableName	nvarchar(128)	The table name where the action happened.
PrimaryKeyField	nvarchar(1000)	The name of the Primary key field in this table.
PrimaryKeyValue	nvarchar(1000)	The value of the Primary key.
FieldName	nvarchar(128)	The field that was changed, inserted, or deleted.
OldValue	nvarchar(1000)	The old value of the field (the first 1000 characters). This is NULL for an Insert operation..
NewValue	nvarchar(1000)	The updated/new value of the field (the first 1000 characters). This is NULL for a delete operation.

DB_INTEGRITY (Deprecated)

DEPRECATED.

DB_SETTINGS

The DB_SETTINGS table stores preferences information that is used by Telescope and I-Pieces. It is, in essence, a registry that can be used to store any information required. This table is maintained internally by Telescope and by individual I-Pieces. The following chart describes the columns in the DB_SETTINGS table.

Field	Data Type	Description
id	identity	Unique ID generated automatically on insert into the table. In Oracle, this is a plain integer column with an insert trigger to populate it from an Oracle sequence.

Field	Data Type	Description
user_name	nvarchar(32)	User name of the user the preference setting belongs to. If this column is NULL, the preference setting is considered to be a "global" preference, available to all users (unless there is a setting with the same keyword for a particular user).
keyword	nchar(32)	Name of the specific preference being described. Each preference setting is a keyword/value pair, so the keyword column is unique for a particular user or group.
valuestr	nvarchar(4000)	The preference setting itself. The contents of this column depend on the keyword associated with it.

Sample DB_SETTINGS Keywords

The following list is a small selection of possible DB_SETTINGs keywords and their values, which appear in the valuestr field.

For more information on using the Solr search keywords, see the *Telescope Administrator's Reference Manual.*

Keyword	Value
disable_search_faceting	When on ("TRUE"), this setting turns off the Refine Search panel (search faceting), with the purpose to improve search query execution times. It is off by default (meaning Refine Search is available by default). To change this default, use "Disable Refine Search" available in TSAdmin's Settings tab. The system checks for this value, and leaves Refine Search on by default if this value is missing or has any value other than "true" or "TRUE". TSAdmin adds this setting with its default (false) value to DB_SETTINGS if it is not found in the table during a settings update when saving the Settings page in TSAdmin.

Keyword	Value
disable_search_highlighting	When on ("TRUE"), this setting turns off search highlighting, with the purpose to improve search query execution times. It is off by default (meaning search highlighting is shown by default). To change this default, use "Disable Search Term Highlighting" available in TSAdmin's Settings tab. The system checks for this value, and leaves search highlighting on by default if this value is missing or has any value other than "true" or "TRUE". TSAdmin adds this setting with its default (false) value to DB_SETTINGS if it is not found in the table during a settings update when saving the Settings page in TSAdmin.
disable_search_weighting	When on ("TRUE"), this setting turns off search relevancy rankings, with the purpose to improve search query execution times. It is off by default (meaning that search weightings are available to users by default). To change this default, use "Disable Search Relevance Weighting," available in TSAdmin's Settings tab. The system checks this value, and leaves ranking on by default if this value is missing or has any value other than "true" or "TRUE". TSAdmin adds this setting with its default (false) value to DB_SETTINGS if it is not found in the table during a settings update when saving the Settings page in TSAdmin.
fr_debug	Determines if debug logging occurs to the DEBUG_LOG table. • If set to "Y" (the default), then log entries will be added to the DEBUG_LOG table • If set to NULL or "N", logging is disabled. To disable logging, issue the following SQL command (for MS SQL): update db_settings set valuestr = 'N' where keyword = 'FR_DEBUG';
ip_vira_open_clip_pl_new_win	When TSWeb users are creating a new playlist (for example, when promoting a clip), this option provides them the option to open and view the new playlist in a separate DocInfo window, if they select a "Open Playlist in new window" checkbox in the confirmation dialog. (This option means they do not need to search for the new playlist after creating it.) Default value is "Y" (on).

Keyword	Value
ip_viravideorendition	For Video Manager files, the type of rendition. 1. VM3 asset (low resolution) 2. Actual file type (high resolution, for example MOOV) 3. QuickFind preference (ip_viravideorendition) (For non-VM files, use the ip_proxyrendition keyword.) For more information on setting up "ip_vira" settings for video ingestion, see the *Video and Audio Transformation Guide*. External storage of proxies: Instead of a rendition number, the value corresponding to this key can contain 'vl_proxies.url'. That is, ip_viravideorendition = 'vl_proxies.url'. This setting will allow for the playback proxy details to come from the VL_PROXIES table. Note that the entry created in the VL_PLAYLISTS table is always populated whenever clips are added to a playlist. If you use a URL, note that the value corresponding to the ip_viravideorendition key is not an integer, and this change may affect existing stored procedures, as well as any triggers on the access_history table that may call these stored procedures. In particular, the following stored procedures may be affected: tsp_createvideoasset, tsp_createpreviewonlyasset, tsp_createplaylistasset. You also need to add the following MIME Type to your IIS configuration for the .qtif (QuickTime Image File) extension: image/x-quicktime Then restart IIS for playback to work.
ip_proxyrendition	For files that are not Video Manager files (that is, MP3, MP4, or FLSH values in the VIEWEX.data_type field), defines the type of rendition 1. VM3 asset (low resolution) 2. Actual file type (high resolution, for example MOOV) 3. QuickFind preference (ip_viravideorendition) For information on external storage of proxies, see the information for ip_viravideorendition above (replacing "ip_videorendition" for "ip_proxyrendition."
isIndexing	This is set to false initially, and set to True only while the Indexing Broker is picking a new batch of record IDs to be indexed. In the unlikely event that two Inexing Brokers are indexing the same database, this flag tries to ensure they aren't processing the same record IDs and are splitting up the indexing job.
lastReindexRecord	The highest record ID value that was re-indexed (generally due to an update to that record).

Keyword	Value
lastSearchBatchIndexRecord	When a batch of record IDs are picked up for processing, the highest record ID number is stored in this value. The Indexing Broker will look for a record ID larger than this number when it assigns the next batch for processing.
lastSearchIndexRecord	The highest record ID value that was actually sent to the Solr Multicore. When the entire batch of record IDs is processed, this value will equal the value of lastSearchBatchIndexRecord (if isIndexing is set to false, this is a good indication that all indexing is complete).
lastSearchIndexTime	This timestamp setting is used internally during Telescope database indexing.
max_failure_count	The maximum number of login failures before a user is locked out. A user is locked out when their login_fail_cnt value (in the users table) matches the max_failure_count value. This value is 99999 by default (a very large number to ensure the user is not locked out).
metadata_cache_timeout	To improve the Solr indexing process, this entry limits the number of database queries issued to validate the data model when indexing an asset. This interval setting is one minute by default (if this entry does not exist) but can be changed by updating the database entry for this keyword with the following command: update db_settings set valuestr='<interval in milliseconds>' where keyword ='metadata_cache_timeout' and user_name is null; Note that this update may require a restart of the broker services.
reindexlastrecord	Triggers automatic reindexing should the network connection be lost "0"-- triggers automatic reindexing when the Child Indexing Broker has its database connection restored. Any other value does not trigger automatic reindexing. By default, this value is not "0" (as of the 9.4.0.7 Telescope release).

Keyword	Value
ui_default_details_view	Configures the Telescope installation so that TSWeb users see alternate views when they first open an asset. To change the popup view, replace the value of valuestr with one of the following: Preview – to show the preview (the default) History – to show the History popup Notes – to show the Notes popup Details – to show the Info popup

DEBUG_LOG

The DEBUG_LOG table is available for logging messages from functional rules, menu functional rules, and other database customizations. It can be populated through custom stored procedures, functions and triggers.

- Debug logging to the DEBUG_LOG table is controlled by the FR_DEBUG entry in the db_settings table.
- Calls to insert data in the DEBUG_LOG table should be done with the tsp_ins_debug_log stored procedure, rather than being inserted directly.
- For more information on using this table, see the section, "Logging Errors from Functional Rules and Customizations" in the *Telescope Administrator's Reference Manual*.

Field	Data Type	Description
id	identity	Unique ID generated automatically on insert into the table. Includes a timestamp.
ENTRY_DATE	datetime (timestamp)	The date of the message entry.
SOURCE	nvarchar(256)	The source name (recommended but optional). If not defined: • In MS SQL, defaults to 'Unknown'. • In Oracle, tries to make a system call to set the name of the calling program.
RECORD_ID	integer	Cross-reference to the asset's record ID in the EDITORIAL table. If not defined, defaults to NULL
MESSAGE_TYPE	nvarchar(1)	The message type. 'D' (for debug), 'E' (for error), or a customized number (as a varchar). If not defined, defaults to NULL.
MESSAGE	nvarchar(max)	The text of the message (as passed by the tsp_ins_error_log variable pc_message).

DL_METHODS

The DL_METHODS table stores the download methods available in the Telescope environment and their associated directories. By default, Telescope downloads files using HTTP. The following chart describes the columns in the DL_METHODS table.

Field	Data Type	Description
id	integer	Unique ID for the method. This is an administrative ID generated automatically by Telescope Admin.
method_dir	nvarchar(255)	Directory (on the web server) where the files downloaded by the method will be copied.
method_name	nchar(32)	Name of the method as it appears to the user.
class_def_yn	nchar(1)	A value of "Y" indicates that the method name refers to a specific class within Telescope. For example, if you set this field to "Y" and the method_name to "QuickLinkSetup", QuickLinks will be enabled as an assignable Download Method.

ERROR_LOG (Deprecated)

This deprecated table was used for logging messages from stored procedures, as passed by the deprecated stored procedure tsp_ins_error_log. Instead, use the DEBUG_LOG table (passed by tsp_ins_debug_log).
For information on Migrating from "error_log" implementations, see "Migration from "error_log" implementations" in the *Telescope Administrator's Reference Manual*.

FM_POLICIES

The FM_POLICIES table stores file migration policies used by the Ingest Broker. The following chart describes the columns in the FM_POLICIES table.

Field	Data Type	Description
fm_name	nchar(64)	The name of the file migration policy.
fm_desc	nvarchar(max)	A description of the file migration policy.

I_PIECES (Deprecated)

DEPRECATED.

JOBS

To ease data entry, Telescope maintains a table of jobs or templates.

Templates created by users in Telescope are stored in the JOBS table and are available for use during data entry.

The following chart describes the columns in the JOBS table.

Field	Data Type	Description
job_name	nvarchar(258)	Name of the job, which will appear in the "Job:" popup menu in the editorial entry dialogs.
job_data	nvarchar(max)	Block of text in MIMiX format that describes the contents of the editorial fields for the job. The contents and format of this field are identical to the contents of the "Editorial Info" file used by Telescope.

LANGUAGE_LOCAL

The LANGUAGE_LOCAL table stores the names for supported languages. It is accessed when the language name needs to be shown in a field called by various functions. This table is referenced from other tables in their lang_id column

to from the values is reserved for future use. Identifies the language (from the language_local table, lang_id column). For example, en_US.

The following chart describes the columns in the LANGUAGE_LOCAL table.

Field	Data Type	Description
lang_id	nchar(10)	The standard ISO identifier for the language. For example, es_ES.
name	nvarchar(128)	Standard English name for this language. For example, "Spanish (Spain)".
local_name	nvarchar(128)	The local name for this language. For example, "Español (España)".
search_code	nchar(2)	The root of the ISO language identifier. For example, "es".
date_format	nvarchar(50)	This field is intended for future use.
number_format	nvarchar(50)	This field is intended for future use.
currency_format	nvarchar(50)	This field is intended for future use.

NAMED_CONV

The NAMED_CONV table contains a list of the defined "named conversions". Named conversions are conversion settings grouped under an identifier. They provide a convenient way for Telescope users to select pre-defined formats for the assets they copy or download. The following chart describes the columns in the NAMED_CONV table.

Field	Data Type	Description
id	integer	A unique key for this table, automatically generated.
conv_name	nvarchar(255)	The name of the conversion as it shows to the user.
conv_types	nvarchar(255)	A comma-separated list of 4-character file types from the database to which this conversion applies.
conv_string	nvarchar(4000)	The conversion string for the named conversion.
conv_factor	integer	The percentage of the original file size which the converted file will be, approximately. In Telescope 9.1 conv_factor is updated to big integer
conv_size	integer	The approximate final size, in bytes, of the converted file. This value is only required if the CONV_FACTOR field is empty. In Telescope 9.1 conv_size is updated to big integer

NPS_DBCHNG_LOG (Internal Use)

This table is used by NorthPlains for internal testing purposes.

RENDITIONS

A rendition is a copy of a file that has the same content but is presented in a different format. A typical usage in Telescope is to render video and audio files during ingest to a specific playback format, also called a proxy.

In the Telescope environment, an individual asset can have multiple renditions. For example, you might have an asset that has three renditions: a high-resolution TIFF image, a medium-resolution JPEG image, and a low-resolution GIF image. Entries into the RENDITIONS table indicate the types of renditions the Telescope application will track for an individual environment.

The following chart describes the columns in the RENDITIONS table.

Field	Data Type	Description

Field	Data Type	Description
rend_id	integer	Unique identifier created by Telescope Admin when the rendition is created.
rend_name	nchar(32)	Name associated with the rendition type used for display purposes.
rend_order	integer	Ordering of the rendition, from 1 .. N, which determines its placement in popup menus, etc. This ordering also determines which rendition is used in a document to generate the thumbnail and extended view for the document.

SEQUENCES

The SEQUENCES table controls the primary key values of tables in the Telescope database by storing the current maximum value of the primary key used by each table. The table stores the current maximum value of the sequence (the record_id) and the table it applies to (sequence_name). These values are sometimes required by the Telescope software.

For example, an entry with record_id = "447778" and sequence_name = "EDITORIAL" indicates that the maximum value of primary keys used in the EDITORIAL table is currently "447778". The SEQUENCES table will contain similar entries for all Telescope tables using primary keys. For example,

368 FORM_SEARCH_FIELDS
18137 access_history
5 announcement_lists
183 announcements
2218 doc_file_info
447778 editorial
And so on …

Whenever Telescope inserts a record into a table with a primary key, it queries the SEQUENCES table to get the current value and then increments that value accordingly. Any customizations that insert data into Telescope data tables must take into consideration the associated sequences and increment them accordingly. Failure to do so will result in Telescope throwing "Duplicate value" exceptions.

The following chart describes the columns in the SEQUENCES table.

Field	Data Type	Description
record_id	integer	Record ID currently in use for the primary key in the table.

sequence_name	nvarchar(32)	Name of the sequence (which is a Telescope table name) for which the database is storing the unique ID.

SES_POOLS

The SES_POOLS table is used to hold the pool names as defined in Telescope Admin. The following chart describes the columns in the SES_POOLS table.

Field	Data Type	Description
id	integer	Unique ID generated automatically on insert into the table. In Oracle, this is a plain integer column with an insert trigger to populate it from an Oracle sequence.
user_class	nchar(2)	The 2-character user class code for the user class of the user.
pool_name	nchar(32)	The name of the pool.

SHARE_MAPPINGS

The SHARE_MAPPINGS table is used by Macintosh software to create FILE_LOCATION information in the DOC_RENDITIONS table on import and to locate files imported by the Windows client. Entries in this table are created using Telescope Administrator. The following chart describes the columns in the SHARE_MAPPINGS table.

Field	Data Type	Description
share_name	nchar(64)	Name of the Windows share as used by the Windows client in the file_location field.
machine_name	nchar(64)	Name of the machine as used by the Windows client in the file_location field.
as_zone	nvarchar(64)	AppleShare zone in which the machine resides as used by the Macintosh client.
as_server	nvarchar(64)	Name of the machine as used by the Macintosh client.
as_path	nvarchar(255)	Path from the root of the shared volume to the shared directory as used by the Macintosh client. Typically, this will be the volume name of the shared AppleShare volume, followed by a colon (e.g., "Volume:").
sh_user	nchar(32)	Default user name used by Telescope to log into this share.
sh_password	nchar(32)	Default password used by Telescope to log into this share.

Field	Data Type	Description
import_flag	nchar(1)	If set to "Y" this share is used for importing and any other value (including NULL) otherwise. At present, this flag is unsupported, but will be used in future releases.
filesys	nchar(10)	Type of file system in use on the network share. Valid values depend on the file system types, but some examples are "NTFS", "MacOS", and "UNIX".

SORTS (Deprecated)

DEPRECATED AND REMOVED.

TS_STATISTICS

The TS_STATISTICS table keeps statistical information about Telescope use.

In very active Telescope environments it is recommended that indexes be added to provide faster access for frequently used statistics. In these environments, the TS_STATISTICS table can also grow very quickly. To prevent database fragmentation and to distribute I/O, it is advisable to physically store the TS_STATISTICS table separately. (In an Oracle environment this can be achieved by using a separate table space. DBAs may want to consider placing the ACCESS_HISTORY and TS_STATISTICS tables together on one table space.) The following chart describes the columns in the TS_STATISTICS table.

Field	Data Type	Description
action_id	identity	Unique ID generated automatically on insert into the table. On Oracle, this is a plain integer column with an insert trigger to populate it from an Oracle sequence.
user_name	nvarchar(32)	User name of the user who performed the action.
action_time	timestamp (datetime)	Timestamp that indicates when the action was performed.

Field	Data Type	Description
action_code	integer	Integer that indicates what action was performed. Possible values are: **1** Login **2** Search **3** Download **4** Deletion **5** Move Files **6** Import **7** File Request **8** Check Out **9** Check In **11** Delete collection (Catalog) **99** Log Out
action_desc	nvarchar(255)	Optional textual description of the action that was performed. For "Deletion" actions, this field contains the name of the file that was deleted. For "Search" actions, it contains the name of the search performed (e.g., "Form" or "Hierarchical"). For searches, the search name is appended to this string, should a name for the search exist. See search_terms for more information on accessing search query data.
action_count	big integer	Contents of this field vary depending on what is in action_code. **Login** – This value is 0. **Search** – The number of documents returned from the search. **Download** – The total number of bytes downloaded. **Deletion** – The total number of documents deleted. **Move Files** – The total number of bytes moved. **Import** – The total number of documents imported. **File Request** – The total number of documents requested. **Check Out** – The number of files checked-out –1, or more through the Download Cart. **Check In** – The number of files checked-in – usually 1. **Log Out** – The number of minutes the user was logged-in. In Telescope 9.1 action_count is updated to big integer
session_id	numeric(19,0)	Unique session ID assigned to the client's session when the user logs in (from the Session Broker).
search_terms	nvarchar(max) (in SQL Server), CLOB (in Oracle)	This column is populated with the search queries entered by users for Simple, Advanced and Form Searches, allowing the use of SQL queries to generate reports from this data. This information is stored as the Solr search string, in Solr format. The search name is appended to the string saved in the action_desc column, should a name for the search exist.

TYPE_CODES

The TYPE_CODES table is used to provide a "real" description of file types as stored in the DOC_RENDITIONS table. Telescope's standard 4-character file type can be confusing to the users, so TYPE_CODES is a "mapping" table that connects the 4-character file types with real names. The TYPE_CODES table also stores a default thumbnail for the file type, which is used if Telescope is unable to generate a thumbnail for the file type. This is useful for file types which do not have graphical representations (such as sound files). The following chart describes the columns in the TYPE_CODES table.

Field	Data Type	Description
type_code	nchar(4)	The 4-character type code, contained in the DOC_RENDITIONS table.
type_name	nchar(256)	A human-readable name that Telescope will use to display the file type in the Document Info window.
default_thumbnail	binary	The encrypted binary data for the default thumbnail to display for this file type, in the event that no thumbnail can be generated from the file's contents by the Graphics Broker during ingest.

Telescope Query Generator

Telescope provides users with the ability to perform ad hoc queries against metadata that resides in several different tables. Generally speaking, Telescope users in a particular environment tend to search the system in well-defined ways. That is, there is usually a set of searches that are performed regularly as part of the workflow. In order to get the most out of the system, these search behaviors should be identified, both through workflow analysis and monitoring the database. To make the most out of a Telescope system, the results of this monitoring should be used to tune the underlying database.

In a system where there is external data referenced (either using a view or metadata residing in custom tables in the Telescope schema), it is not uncommon for a user to enter a criteria that causes a search across three or more tables: EDITORIAL, DOC_RENDITIONS, and one or more custom tables or views. Telescope performs the search using an outer join between the tables using the following format:

```
select RECORD_ID from EDITORIAL e, DOC_RENDITIONS d, CUSTOM_TABLE d1 where
e.RECORD_ID = d.RECORD_ID and e.RECORD_ID = d1.RECORD_ID and (user's where
clause) and (user's search criteria);
```

The Telescope application always assumes that the search should be case-insensitive. This is achieved through the use of the RDBMS UPPER function. For example, if the user searches for a file named "rose.jpg", the query would include:

```
and upper(d.FILE_NAME='ROSE.JPG')
```

A standard index on the FILE_NAME column on some databases would not assist in the performance of this query. Alternative tuning methods should be considered, such as the possible use of a functional rule in Oracle. Depending on the RDBMS, adding constraints such as primary and foreign keys may assist with the execution of the query.

Appendix: Programmability

WARNING: Do not delete, disable, or modify any Telescope Function, Stored Procedure, Triggers, or Views.

Functions and Stored Procedures

tsp_acquirecheckoutlock

Database Type	Programmability
SQL Server	Procedure
Oracle	Function

Calls the tsp_acquirecheckoutlock_impl procedure to lock an asset so it cannot be checked out by another user.

`tsp_acquirecheckoutlock(record_id)`

`record_id`: The record ID of the asset being checked out.

`Returns`: The return code from the tsp_acquirecheckoutlock_impl procedure.

tsp_acquirecheckoutlock_impl

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Marks a record as checked out so other users cannot check out the file.

`tsp_acquirecheckoutlock_impl(record_id, ret)`

`record_id`: The record ID of the asset being checked out.

`ret`: The return code as follows:

0 – if the file was successfully checked out.

1 – if the file is already checked out.

-1 – if file was not found.

`Returns`: See "ret" above.

tsp_add_setting

Database Type	Programmability
SQL Server	Procedure
Oracle	Function

Sets a db_settings table entry.

`tsp_add_setting(user_name, keyword, valuestr)`

`user_name`: The username or group name to which this value applies.

`keyword`: The name of the keyword.

`valuestr`: The value to set.

Returns: 0

tsfn_charindexr

Database Type	Programmability
SQL Server	Function
Oracle	N/A

Utility function to search for a pattern in a string, starting from the end.

`tsfn_charindexr(expression1, expression2, start_location)`

`expression1`: The pattern to search for.

`expression2`: The string to search in.

`start_location`: The position in the string to start the search from. By default, the search is started from the end.

`Returns`: The position of the pattern you searched for in the string.

tsp_createMetadataField

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Creates metadata fields.

tsp_createMetadataField (column_name, table_name, column_display, datatype, dbdatatype, length, isrequired, issearchon, isfaceton, isvalidate, description, popup, isCascade, cascadeColumn, cascadeValues, bucketType, bucketSize, bucketStart, bucketEnd, bucketStartInt, bucketEndInt, bucketTimeRange, isSkip);

column_name: The name of the Field, e.g.: "asset_type"

table_name: The name of the table.

column_display: The Display Name and localization for that name(in xml fomat), e.g.: "<DISPLAYNAME><LOCAL NAME="default">Asset Type</LOCAL></DISPLAYNAME>"

datatype: A number that represent a data type in Telescope

dbdata_type: The type of the field

length: The maximum length for the values

isrequired: If the field will be required, supported values: 'Y', 'N'

issearchon: If the field will be search on, supported values: 'Y', 'N'

isfaceton: If the field will be facet on, supported values: 'Y', 'N'

isvalidate: If the field will be validated, supported values: 'Y', 'N'

description: The description of the field

popup: The Popup values whith localization(in xml fomat), e.g.: "<POPUP><LOCAL NAME="default">|value1|value2|</LOCAL></POPUP>"

isCascade: If the field has a cascade relation to display it, supported values: 'Y', 'N'

cascadeColumn: The name of the field with the cascade raltion

cascadeValues: The values of the popup that will used to show this field

bucketType: The type of the bucket (Values, Range)

bucketSize: The size of the bucket

bucketStart: The start range (Date format)

bucketEnd: The end range (Date format)

bucketStartInt: The start range (Integer format)

bucketEndInt: The end range (Integer format)

bucketTimeRange: The range type if the range is for date (Seconds, Minutes, Years,etc)

isSkip: If the field already exist will be skipped, supported values: 'Y', 'N'

Returns: N/A

tsp_createMetadataSmartCatalog

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Used by tstrg_embedded_metadata_ins to create new smart collections (catalogs) based on the tag information returned from the Metadata I-piece. See also tsp_deleteMetadataSmartCatalog.

```
tsp_createMetadataSmartCatalog(tag)
```

`tag`: The new collection name.

Returns: N/A

tsp_createNRtable

Database Type	Programmability
SQL Server	Procedure
Oracle	Function

Creates a table necessary for normalized repeating fields. The table name is the same as the column name specified, with the prefix "nr_" added. This function is used by internal North Plains tools to populate Telescope environments with metadata.

tsp_ createNRtable (column_name, column_display, length, isrequired, issearchon, isfaceton, isskip)

column_name: The name of the column (nvarchar2). The name of the table created is the same as this name, with the prefix "nr_" added.
column_display: The display name and localization for that name (in XML format), varchar. For example: "<DISPLAYNAME><LOCAL NAME="default"> Visible to Departments</LOCAL></DISPLAYNAME>"
length: The length of the column (nvarchar2, must be a number ranging from 1 to 255).
isrequired: Specifies if the column is required ('Y' or 'N', default is 'N', no)
issearchon: Specifies if users can search on contents in the column ('Y' or 'N', default is 'N', no)
isfaceton: Specifies if the column will be included in the Refine Search panel ('Y' or 'N', default is 'N', no)
isskip: Specifies if the column should be skipped ('Y' or 'N', default is 'N', no)

Returns: 0 if successful, -1 if not successful.

Example:
exec tsp_createNRtable 'nr_vis_department', '<DISPLAYNAME><LOCAL NAME="default">Visible to Departments</LOCAL><LOCAL NAME="fr_CA">Visible aux ministères</LOCAL><LOCAL NAME="es_ES">Visible a Departamentos</LOCAL></DISPLAYNAME>', '150', 'N', 'Y', 'Y', '';

tsp_createplaylistasset

Database Type	Programmability
SQL Server	Procedure

Oracle	Procedure

Promotes an asset to a playlist by setting the asset's data_type in the VIEWEX table to 'TPLT' and creating an entry for the asset in the DOC_RENDITIONS table.

`tsp_createplaylistasset(record_id)`

`record_id`: The record ID of the asset being promoted to a playlist.

Returns: N/A

tsp_createplaylistfromclip

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Creates a "clip asset" from a selected clip in the Video Manager interface.

tsp_createplaylistfromclip (record_id, inoutmsec, playlist_name)

record_id: The record IDof the asset being checked out.

inoutmsec: The in and out values (in milliseconds), in the following format: "<inmsec>:<outmsec>"

playlist_name: The field the playlist name should be put into.

Returns: 0 if successful, -1 with error message if not successful.

tsp_createpreviewonlyasset

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Designates an asset as a "preview-only" asset that will open in the default preview by setting the asset's data_type in the VIEWEX table to the file type of the asset and creating an entry for the asset in the DOC_RENDTIONS table.

```
tsp_createpreviewonlyasset(record_id)
```

`record_id`: The record ID of the asset being created.

Returns: N/A

tsp_createTableExtendEditorial (Internal Use)

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

This procedure is used by internal tools to create tables linked to the EDITORIAL table and to extend the metadata model.

tsp_createvideoasset

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Designates an asset as a video to be edited in Video Manager 3.0 by setting the asset's data_type in the VIEWEX table to 'ViRa' and creating an entry for the asset in the DOC_RENDTIONS table.

```
tsp_createvideoasset(record_id)
```

`record_id`: The record ID of the asset being created.

Returns: N/A

tsp_delete_record

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Properly removes a record and its access history from the database.

`tsp_delete_record(record_id)`

`record_id`: The record ID of the asset being deleted.

Returns: N/A

tsp_deleteMetadataSmartCatalog

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Deletes a smart collection (catalog).

`tsp_deleteMetadataSmartCatalog(tag)`

`tag`: The catalog name to be deleted.

Returns: N/A

tsp_delete_version

Database Type	Programmability
SQL Server	Procedure

Oracle	Procedure

Properly removes a version from the database.

`tsp_delete_version(version_id)`

`version_id`: The version ID being deleted.

Returns: N/A

tsp_FindAllChildren

Database Type	Programmability
SQL Server	Procedure
Oracle	Function

Finds all placed files for the Collect function in the Component Object View.

`tsp_FindAllChildren(record_id, rend_ids, where_clause)`

`record_id`: The record ID of the parent asset.

`rend_ids`: The rendition IDs of the parent asset.

`where_clause`: A where clause to exclude assets not allowed by the security model.

`Returns`: A table containing the record id, rend id, and file checksum for each record found.

tsp_FindAllParents

Database Type	Programmability
SQL Server	Procedure
Oracle	Function

Finds all assets a placed file is defined as a child of (i.e. assets that have the file listed in a container field).

```
tsp_FindAllParents(record_id, rend_id, where_clause)
```

`record_id`: The record ID of the child asset.

`rend_id`: The rendition ID of the child asset.

`where_clause`: A where clause to exclude assets not allowed by the security model.

`Returns`: A table containing the record id, rend id, and file checksum for each record found.

tsp_FindChildAsset

Database Type	Programmability
SQL Server	Procedure
Oracle	Function

Finds the asset when a user clicks a placed file in the Component Object View.

```
tsp_FindChildAsset(record_id, page_num, geometry_order, where_clause)
```

`record_id`: The record ID of the child asset.

`page_num`: The page number of the asset.

`geometry_order`: The geometry order of the asset.

`where_clause`: A where clause to exclude assets not allowed by the security model.

`Returns`: A table containing the record id, rend id, and file checksum for the record.

tsfn_getcontainers

Database Type	Programmability
SQL Server	Function
Oracle	Function

For Advanced Search, includes "Member of" container fields for the specified user. Fields are filtered based on the passed-in user's group's field visibility.

```
tsfn_getcontainers(vUserName)
```

`vUserName`: A user name (nvarchar(32)). The user name of the user requesting the Advanced Search

Returns: The column ID and name of all the container fields the user has permission to see (nvarchar(max)). This data is JSON-encoded and in the format [{"id": 123, "name": "abcde"}, ...] Fields will be filtered based on the passed-in user's group's field visibility.

tsp_getfieldvalue

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Called by other procedures, this procedure gets the value of a field in a given table for a given record ID.

```
tsp_getfieldvalue(record_id, table_column_name, value)
```

`record_id`: The record ID of the asset.

`table_column_name`: The table column name.

`value`: This is an 'out' field that will contain the requested field.

`Returns`: The 'value' field above.

tsp_getfiletypes

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Internal procedure used to get a list of current file types, used for advanced searches. There are no parameters for this procedure. Returns a JSON-encoded string consisting of the file_types from the TYPE_CODES table.

tsfn_getfiletypes

Database Type	Programmability
SQL Server	Function
Oracle	Function

Internal function used to get a list of current file types, used for advanced searches. There are no parameters for this function. Returns a JSON-encoded string consisting of the file_types from the TYPE_CODES table.

tsp_getMimix

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Returns metadata information in MIMiX format for an asset with a specified record ID.

```
tsp_getMimix(record_id)
```

`record_id`: The record ID of the asset being requested.

getnegativeExtraColumnsID (Internal Use)

This procedure is used by internal tools. It calls the tsp_getnextid_impl procedure to get the next available number in a sequence from the SEQUENCES table.

tsp_getnextid_impl

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Gets the next available number in a sequence, in order to provide a unique ID for tables requiring it.

Some tables require unique IDs to identify new entries, which are generated by incrementing the latest value, stored in the SEQUENCES table. This procedure increments the latest number in this table to provide a unique ID. If the table you are requesting the ID for does not have a record in the SEQUENCES table, this procedure creates one.

```
tsp_getnextid_impl(sequence_name, column_name, sequence_id, number_to_allocate,
sequence_direction)
```

`sequence_name`: The name of the sequence (which is a Telescope table name).

`column_name`: The name of the column containing the value the ID is for.

`sequence_id`: This is an 'out' field that will contain the current max value of the primary key for the table you are searching on

`number_to_allocate`: The number of ids to allocate to the sequence (for batch operations). This number is added to the current max value and set as the new next available ID.

`sequence_direction`: The direction of the sequence; pass "1" to increase the values, pass "-1" to decrease the values.

`Returns`: The sequence_id above.

For more information about using sequences, see "SEQUENCES" on page 129.

tsfn_getpopups

Returns popups in JSON-encoded format.

tsfn_jsonencode(vUserName)

vUserName: Used to determine the language of the popups to return

Returns: A string of the popups as JSON-encoded characters.

tsp_getpopups

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Returns a JSON encoded list of popup items based on the user's permissions.

```
tsp_getpopups (vUserName)
```

`vUserName`: The user name of the current user.

Returns: A string listing the popups, in JSON-encoded characters.

tsp_getplaylistassetname

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Because playlist assets are metadata-only assets that act as containers for clips from other assets, they do not have a file name or other obvious identifier associated with them. To make it easier for users to distinguish between playlist assets, they may create a metadata field and designate it as the playlist "name." This procedure gets the value of that field for a given record ID.

```
tsp_getplaylistassetname(record_id, value)
```

`record_id`: The record ID of the asset.

`value`: This is an 'out' field that will contain the name of the playlist asset defined in the db_settings table.

`Returns`: The 'value' field above.

tsp_getsetting

Database Type	Programmability
SQL Server	Function
Oracle	Function

Gets the name of the FlipFactory I-Piece queue to process an imported video asset with.

```
tsp_getsetting(user_name, keyword)
```

`user_name`: The user name of the default import user (as defined in the com.northplains.ipiece.flipfactory.xml file).

`keyword`: The keyword of the entry in the DB_SETTINGS table that defines the queue to use for this user. The tsp_ipflip_getimportqueue procedure passes 'ipflip_importqueue' by default.

`Returns`: The name of the FlipFactory queue.

tsp_getvideoassetdescriptor

Database Type	Programmability
SQL Server	Procedure
Oracle	Function

Used by the trigger on the ACCESS_HISTORY table to determine whether a new or updated asset is a playlist, a video or a preview-only asset.

```
tsp_getvideoassetdescriptor(record_id, type)
```

`record_id`: The record ID of the asset being promoted to a playlist.

`type`: The asset type as passed by the trigger on the ACCESS_HISTORY table.

Returns: 0

tsp_ins_debug_log

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Calls to insert data in the DEBUG_LOG table should be done with the TSP_INS_DEBUG_LOG stored procedure, rather than being inserted directly. The TSP_INS_DEBUG_LOG method signature (in both SQL Server and Oracle) is:

(exec) tsp_ins_debug_log <Message>, <Source>, <Type>, <Record_id>, <Debug Y/N>

Where:

- <Message> is the message text (required).
- <Source> is the source name (recommended but optional). If not defined:
    - In MS SQL, defaults to 'Unknown'.
    - In Oracle, tries to make a system call to set the name of the calling program.
- <Type> is the message type. 'D' (for debug), 'E' (for error), or a customized number (as a varchar). If not defined, defaults to NULL.
- <Record_id> is the asset record ID (record_id). If not defined, defaults to NULL.
- <Debug Y/N> is a flag to indicate whether or not the stored procedure should make a log entry. If set to Y, it will always log, say for a critical error (even if the "FR_DEBUG" value in the db_settings table is set to N). If set to N, it will not make a log entry (even if the "FR_DEBUG" value in the db_settings table is set to Y). If this flag is not passed, the procedure will query the "FR_DEBUG" value in the db_settings table to determine whether or not to log.

NOTE: The order of these parameters has changed from those of most of the preceding tsp_ins_error_log method, in order to put required parameters first. For information on Migrating from "error_log" implementations, see "Migration from "error_log" implementations" in the *Telescope Administrator's Reference Manual*.

tsp_ins_error_log (Deprecated)

This deprecated procedure was used for logging messages to the deprecated ERROR_LOG table.

Instead, use the DEBUG_LOG table (passed by tsp_ins_debug_log). For information on Migrating from "error_log" implementations, see "Migration from "error_log" implementations" in the *Telescope Administrator's Reference Manual*.

tsp_ipflip_getimportqueue

Database Type	Programmability
SQL Server	Procedure
Oracle	Function

Calls the tsp_getsetting function to get the name of the FlipFactory I-Piece queue to process an imported video file with.

```
tsp_ipflip_getimportqueue(record_id, rend_id, user_name, file_info)
```

`record_id`: *DEPRECATED.*

`rend_id`: DEPRECATED.

`user_name`: The user name of the default import user (as defined in the com.northplains.ipiece.flipfactory.xml file).

`file_info`: *DEPRECATED.*

`Returns`: The name of the FlipFactory I-Piece queue.

tsfn_jsonencode (Internal Use)

This is an internal function used for JSON encoding.

tsfn_MsecToSmpte

Database Type	Programmability

SQL Server	Function
Oracle	Function

Converts the video timecode from milliseconds to an SMPTE string.

```
tsfn_MsecToSmpte(msec, frame_rate)
```

`msec`: The video timecode in milliseconds.

`frame_rate`: The video frame rate.

`Returns`: The video timecode in a SMPTE-formatted string (in the format hh:mm:ss:frame).

tsp_newdocument

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

This is a sample procedure to show the use of Telescope procedures.

```
tsp_newdocument(record_id)
```

`record_id`: This an 'out' field that will contain the ID of the new document created.

`Returns`: The 'record_id field above.

tsp_parse_str

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Parses the content and returns the parsed values in a string.

`tsp_parse_str(str, delimiter)`

`str`: The string to be parsed.

`delimiter`: The delimiter to parse by.

`Returns`: The result of the parse.

tsp_popularFeed, tsfn_popularFeed

Database Type	Programmability
SQL Server	Procedure (tsp_popularFeed)
Oracle	Procedure (tsfn_popularFeed)

Retrieves and returns (as a JSON array) the 10 most popular assets in the database (based on the number of times the assets have been downloaded).

`tsp_popularFeed(user_name)`

`user_name`: A user name (nvarchar(32), optional). If present, the user's where clause is used to compile the top ten list. If not present, the top ten is compiled without any where clause.

`Returns`: A JSON array

tsp_recentFeed, tsfn_recentFeed

Database Type	Programmability
SQL Server	Procedure (tsp_recentFeed)

Oracle	Procedure (tsfn_recentFeed)

Retrieves and returns (as a JSON array) the 10 most recent assets in the database based on the access-history for access type 4

`tsp_recentFeed(user_name)`

`user_name`: A user name (nvarchar(32), optional). If present, the user's where clause is used to compile the top ten list. If not present, the top ten is compiled without any where clause.

`Returns`: A JSON array

tsp_setupLanguages

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Sets up default and supported languages.

tsp_setupLanguages (default_language, language_ids)

default_language: The code of the default language, e.g.: "en_US"

language_ids: The list of the supported languages separated by a comma, e.g.: "en_US,fr_CA,es_ES"

Returns: Return code (Integer)

tsfn_SmpteToMsec

Database Type	Programmability
SQL Server	Function
Oracle	Function

Converts the video timecode from SMPTE bits to milliseconds.

```
tsfn_SmpteToMsec(smpte, rate)
```

`smpte`: The video timecode in SMPTE bits (in the format hh:mm:ss:frame).

`rate`: The video frame rate.

`Returns`: The video timecode in milliseconds.

tsfn_toUnixTimestamp

Database Type	Programmability
SQL Server	Function
Oracle	Function

Converts a date to a UNIX timestamp integer.

tsfn_toUnixTmestamp(datetime)

datetime: can be datetime or date

Returns: Unix timestamp integer

tsp_update_md5

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Used to update the MD5 (checksum) key for import and download to/from the Xinet File Broker.

```
tsp_update_md5(record_id, rend_id, MD5_key)
```

record_id: The record ID of the asset.

rend_id: The rendition ID of the asset.

MD5_key: The MD5 checksum hash key.

Returns: integer. (return code?)

tsp_vm3_repairinfo

Database Type	Programmability
SQL Server	Procedure
Oracle	Procedure

Updates the duration and frame rate in the vl_info table for a video asset. It also sets the tape offset value to 0 for the same asset record.

```
tsp_vm3_repairinfo(pRecordID, pDuration, pFrameRate)
```

`pRecordID`: The record ID of the video asset being updated.

`pDuration`: The new value for duration.

`pFrameRate`: The new value for the frame rate.

Returns: N/A

Functional Rules

tsfr_ApplyOfficeMetadata (Deprecated)

Deprecated

tsfr_ApplyXMPMetadata

Database Type	Programmability

SQL Server	Procedure
Oracle	Function

Called by the Telescope conversion functional rules to apply metadata when a user downloads a file processed by the XMP Conversion I-Piece. The file type is checked within this functional rule procedure against a list of supported file types. (For details on how to update this list, see the *XMP I-Piece and Conversion I-Piece Manual.)*

`tsfr_ApplyXMPMetadata(record_id, rend_id, final_file_type)`

`record_id`: The record ID of the asset being downloaded.

`rend_id`: The rendition ID of the rendition being downloaded.

`final_file_type`: The final file format of the file being downloaded (this may be the original file type, or the end result of a conversion string applied to the file when it is downloaded).Returns: The conversion string.

tsfr_ApplyPlayListMetadata

Database Type	Programmability
SQL Server	Procedure
Oracle	Function

Called by the Telescope conversion functional rules to apply metadata when a user downloads a playlist asset.

`tsfr_ApplyPlayListMetadata(record_id, rend_id, final_file_type)`

`record_id`: The record ID of the asset being downloaded.

`rend_id`: Deprecated.

final_file_type: *DEPRECATED.*

`Returns`: The conversion string.

tsfr_GetIndesignConnectInfo

Database Type	Programmability
SQL Server	Procedure
Oracle	Function

Called by the Telescope conversion functional rules to apply metadata when a user downloads an Adobe InDesign asset.

Triggers

trig_editorial_approval_update

A trigger on the EDITORIAL table that inserts a row into the SEARCH_INDEX_ACTIONS table to re-index the asset when the APPROVPEND flag is updated from Y to null.

tstrg_access_history_ins

A trigger on the access_history table (on insert) that calls tsp_createplaylistasset, tsp_createvideoasset, or tsp_createpreviewonlyasset.

tstrg_cov_info_ins

A trigger on the cov_info table (on insert) that sets the dflt_display to 'THUMBS' for PowerPoint documents

tstrg_embedded_metadata_del

A trigger on the embedded_metadata table (after delete) that calls the tsp_deleteMetadataSmartCatalog for each row in the table.

tstrg_embedded_metadata_ins

A trigger on the embedded_metadatatable (after insert) that that calls the tsp_createMetadataSmartCatalog for each row in the table.

tstrg_extra_columns_ins

A trigger on the extra_columns table (after insert) that sets the distribute_yn flag to `Y`. When the Distribution Broker is configured to distribute the metadata for an asset in a MIMIX file, or when using any metadata field for file or directory name substitution, the Distribution Broker will only pickup fields with distribute_yn set to 'Y'. If a field does not have that flag set, that field will not get distributed.

tstrg_vl_info_insupd

A trigger on the vl_info table(after insert or update) that recalculates and sets the in_smpte in the vl_clips, vl_text, and vl_thumbnails tables. It also updates last_update in the vl_info table to the current system date.

tstrg_vl_playlist_url_ai

A trigger on the vl_info table that adds the needed data for MassStore Broker when a new playlist is created.

TSTRG_USER_NAME_HISTORY

A trigger on the USER_NAME_HISTORY table that records user history: insertion and updates to users.

Audit Table Triggers

Each of the following tables has a trigger that updates the AUDIT table with the action taken.

Table Name	Trigger Name
announcement_list_groups	announcement_list_groups_ctts
announcement_list_moderators	announcement_list_moderators_ctts
announcement_lists	announcement_lists_ctts
announcements	announcements_ctts

checkouts	checkouts_ctts
cov_majortypes	cov_majortypes_ctts
cov_sectiontypes	cov_sectiontypes_ctts
db_settings	db_settings_ctts
db_integrity	db_integrity_ctts
dl_methods	dl_methods_ctts
ed_versions	ed_versions_ctts
extendedview_fields	extendedview_fields_ctts
extra_columns	extra_columns_ctts
fm_policies	fm_policies_ctts
fn_messages	fn_messages_ctts
fn_rules	fn_rules_ctts
fn_rulesets	fn_rulesets_ctts
fn_watermarks	fn_watermarks_ctts
form_search	form_search_ctts
form_search_fields	form_search_fields_ctts
form_search_values	form_search_values_ctts
hier_items	hier_items_ctts
hier_levels	hier_levels_ctts
hierarchies	hierarchies_ctts
iconic_fields	iconic_fields _ctts

jobs	jobs_ctts
m_actions	m_actions_ctts
m_attachments	m_attachments_ctts
mmx_sync	mmx_sync_ctts
named_conv	named_conv_ctts
paraview_fields	paraview_fields_ctts
popups	popups_ctts
renditions	renditions_ctts
ses_pools	ses_pools_ctts
share_mappings	share_mappings_ctts
textview_fields	textview_fields_ctts
type_codes	type_codes_ctts
users	users_ctts
welcome_iconic_levels	welcome_iconic_levels_ctts
welcome_iconic_searches	welcome_iconic_searches_ctts
welcome_icons	welcome_icons_ctts
welcome_pages	welcome_pages_ctts
ws_archive	ws_archive_ctts
ws_decisions	ws_decisions_ctts
ws_j_notifications	ws_j_notifications_ctts
ws_j_users	ws_j_users_ctts

ws_junctions	ws_junctions_ctts
ws_rm_notifications	ws_rm_notifications_ctts
ws_routemaps	ws_routemaps_ctts
ws_s_notifications	ws_s_notifications_ctts
ws_sj_notifications	ws_sj_notifications_ctts
ws_sj_users	ws_sj_users_ctts
ws_st_users	ws_st_users_ctts
zoom_info	zoom_info_ctts

Views

tsvw_doc_renditions

A view on the editorial and doc_renditions tables defined as:

```
select e.record_id, count(d.record_id) as rend_count from editorial e left join
doc_renditions d on e.record_id = d.record_id group by e.record_id
```

tsvw_embedded_metadata

A view on the embedded_metadata table defined as:

```
select record_id, ('<table><tr><td  style="width:400px;font-weight:bold;text-
align:right;padding-right:20px">' + tag + '</td><td>' + value +
'</td></tr></table>') as tag_value from embedded_metadata
```