TeleScope Design Documentation

I-Piece Design and API

Revision 1.5

1-July-1996

Table of Contents

1. Overview 1

    I-Piece Files 1

    I-Piece Signatures 1

    I-Piece Distribution 1

2. I-Piece Format 2

    Macintosh 2

    Windows 2

3. Usage Summary 3

    Who Calls I-Pieces? 3

    Priority 3

    When Are They Called? 3

        Startup 3

        File Acquisition 4

        Viewex Display 4

        Preferences Display 4

        Copying/Moving Files 5

        Shut Down 5

4. TeleScope Runtime Support 6

    Graphics Support 6

    Keyword Support 6

    Global Variables 6

    Preferences 6

    Time Sharing 7

    Resources (Macintosh) 7

Code Movement (Macintosh) 7
5. Application Programming Interface 8
Calling Conventions 8
Return Status 8
IPInitialize 9
IPSetPrefs 10
IPAcquireFile 11
IPClaimDisplay 12
IPDrawDisplay 14
IPDisplayEvent 14
IPCloseDisplay 15
IPDisplayPrefs 16
IPGetPrefs 17
IPMoveFile 18
IPClose 19
6. Data Structures 20
IPFileInfo Record 20
IPDisplayInfo Record 27
IPMoveInfo Record 30
Appendix A: Developer Support 32

1. Overview

TeleScope in designed to be as independent of individual file formats as possible. To this end, a plug-in architecture known as **I-Piece Technology** is used within TeleScope to recognize, parse, and display to the user different file formats. Using this method, new or proprietary file formats can be supported within TeleScope simply by plugging in an I-Piece designed for that format.

I-Piece Files

I-Pieces are distributed in **I-Piece Files** which contain both the code and resources required for an I-Piece to function. Since portions of the TeleScope software run on both the Macintosh and Windows platforms, there are two types of I-Piece Files. On Macintosh, I-Pieces are code resource files, and on Windows, they are DLL files.

I-Piece Signatures

Each I-Piece is assigned a unique signature number, to distinguish it from other I-Pieces. North Plains Systems Inc. maintains a registry of these signatures, and third-party developers of I-Pieces should obtain a unique signature before shipping their product (see the contact information in *Appendix A: Technical Support*). The signature given an I-Piece also indicates its priority relative to other I-Pieces installed on a user's system (see the section entitled *Priority* in this document). In Macintosh I-Piece files, this signature is contained in the **IPpr** resource. On Windows, the I-Piece's signature is embedded in the DLL's version information.

I-Piece Distribution

Unlike most plug-in technologies, I-Pieces must be distributed in a client/server environment. With a typical plug-in architecture, new plug-ins must be installed by hand on each computer which has the software installed. Because TeleScope is designed for large-scale client/server implementation on local or wide-area networks, such a method for distributing I-Pieces would be undesirable. Thus, TeleScope uses an automatic distribution method whereby I-Pieces are installed once, by the System Administrator, and are stored at the server, in the TeleScope database. When TeleScope client software (either Engine, QuickDrop, or Viewer) is launched, the I-Pieces in the database are compared against any current I-Pieces which are resident on the client computer. Any new or changed I-Pieces are automatically downloaded from the database and installed in the appropriate system directories on the client computer. This allows the System Administrator to automatically install new I-Pieces, or update existing ones, in one step. The TeleScope software itself guarantees that the new I-Pieces will be propagated to each machine on which it is required.

2. I-Piece Format

Once it has downloaded an I-Piece from the database, the TeleScope software will store it in a platform-specific format, as defined below:

Macintosh

On the Mac, I-Pieces are stored in the TeleScope preferences folder, at the following location:

```
<Volume>:System Folder:Preferences:TeleScope ƒ:I-Pieces:
```

The I-Piece file name is retrieved from the database, and the file is created with a creator code of either TeVi or TeEn (depending on the application which downloaded it) and a file type code of IPRF.

Contained in the resource fork of an I-Piece File, there must be one of each of the following types of resources:

IPpr	There must be only one 'IPpr' resource (with any resource ID) in the I-Piece file. The 'IPpr' resource data is a binary 4-byte integer (represented in Motorola byte ordering). This integer is the I-Piece's unique priority signature, as described above.
IPcc	There must be only one 'IPcc' resource in the file, with any resource ID. It is a Macintosh 680x0 code resource, which will be the main calling point for Macintosh TeleScope software running on 68k Macintoshes. If the IPcc resource is not present, then the I-Piece contained in this I-Piece File will not be usable on a 68k Macintosh computer.
IPpc	There must be only one 'IPpc' resource in the file, with any resource ID. It is a Macintosh PowerPC code resource, which will be the main calling point for PowerMac TeleScope software. If the IPpc resource is not present, then the I-Piece contained in this I-Piece file will not be usable on a PowerMac computer.
cicn	The 'cicn' resource with resource ID 128 is used in the TeleScope software to display an icon for the I-Piece. This icon should be 16 x 16 pixels in size to display correctly.

Windows

On Windows, the I-Piece is stored in the Windows system directory where globally-accessible DLL's reside. The file name is retrieved from the database.

3. Usage Summary

Who Calls I-Pieces?

I-Pieces are used by all of the TeleScope client modules (Engine, QuickDrop and Viewer). Unlike other plug-in application technologies (like Quark XTensions, Photoshop Plugins or HyperCard XCMD's) where a specific extension is called for a specialized purpose, in TeleScope, all installed I-Pieces are called, in sequence, for most operations the TeleScope software performs. Each I-Piece has an opportunity to respond to the call or pass it on to the next I-Piece for processing. I-Pieces accomplish this by returning a status value to the TeleScope software, indicating whether the I-Piece has handled the call, and whether the TeleScope software should continue down the chain of installed I-Pieces or immediately halt. This process is covered in detail in the next sections.

Priority

As mentioned previously, each I-Piece has a unique identifying signature, which is stored in the I-Piece file. This signature is an integer value, which also represents the I-Piece's priority within TeleScope. The TeleScope software will call I-Pieces with lower priority values before those with higher priority values.

North Plains Systems Inc. reserves all the following priority numbers (signatures) for its own use: 0 to 255, 1,073,741,696 to 1,073,741,951, and 2,147,483,392 to 2,147,483,647. This allows North Plains to issue I-Pieces which are guaranteed to be called first, last, and somewhere in the middle of the calling sequence.

As an example of the prioritized calling of I-Pieces, consider the *Keyword* I-Piece, supplied with the TeleScope software by North Plains Systems Inc. The *Keyword* I-Piece has a priority of 2,147,483,647, ensuring that it will be called last. This is required for this I-Piece, because it parses the textual data returned by other I-Pieces, and retrieves keywords from them to store in the *keywords* table in the database. Thus, regardless of what type of file is being processed, the *Keyword* I-Piece will be able to extract information from its textual data, after it has been processed.

When Are They Called?

I-Pieces are called by the TeleScope Engine and Viewer software at six well-defined times:

Startup

When the TeleScope software starts up, it loads all the I-Pieces, and calls an initialization routine in each of them, which allows the I-Piece a chance to allocate any working storage that it needs. The TeleScope software calls each I-Piece after it is loaded, as follows:

IPInitialize to allow the I-Piece to allocate any memory it may need.

IPSetPrefs if the TeleScope software has a preferences record for the I-Piece, it will pass it. Otherwise (if, for example, the user has never displayed the preferences dialog for a particular I-Piece) this call will not be made.

File Acquisition

Whenever a file needs to be processed into the database, whether it is because the user has dragged a file into the TeleScope Viewer or QuickDrop software to initiate an import operation, or the TeleScope Engine determines that a new file has been dropped into one of its monitored folders, the TeleScope software uses the I-Pieces to help process the file into the database. To do this, the software opens the file, extracts any file-specific information from the file, and places it into an *IPFileInfo* structure. Each I-Piece's *IPAcquireFile* routine is then called in priority order, passing this structure into the I-Piece, which may modify the data contained in the structure if appropriate before returning. Upon return from the call, the I-Piece may indicate that it modified the data in the structure, and additionally control the order in which the TeleScope software calls other I-Pieces.

Viewex Display

When the user of the TeleScope Viewer software activates the viewex view, the Viewer loads an *IPDisplayInfo* structure with the viewex data from the database, and various internal information. The *IPClaimDisplay* routine of each I-Piece is then called in priority order, until one of them indicates that it can handle the viewex data, and the Viewer stops calling I-Pieces. The I-Piece which responded to the claim request is said to "own" the viewex view. The owning I-Piece is then called using the *IPDrawDisplay* routine, to actually draw the viewex data in the window which has been created. Additionally while the viewex view is on the screen, the Viewer software calls the *IPDisplayEvent* routine of the I-Piece which owns the view each time the user generates an event (mouse down, or key down, for example) inside the viewex view. Finally, when the user closes the viewex view, the Viewer software informs the I-Piece using the *IPCloseDisplay* call.

Preferences Display

In certain cases, the user may wish to alter the operation of individual I-Pieces. To provide for this functionality, the TeleScope software can request an I-Piece to display a preferences dialog specific to itself, and allow the user to modify internal settings using this interface. In the Engine, these preference settings will affect automatic file acquisition to the database. In the Viewer, these settings will affect the import/display operation of the I-Piece for that Viewer. The TeleScope software calls the appropriate I-Pieces as follows:

IPGetPrefs when the user selects an I-Piece's icon in the preferences window, the TeleScope software asks the selected I-Piece for its current preferences information using this call.

IPDisplayPrefs the TeleScope software then calls the I-Piece to display its preferences dialog, passing in the preferences information it

received from the previous call to *IPGetPrefs*. The I-Piece should use this preferences block to display in the dialog, and modify as the user changes values. It is passed back to the TeleScope software when the dialog is dismissed by the user.

u Important

The I-Piece should *not* set it's internal preferences at this time, because the user may still cancel out of the I-Piece preferences window in the TeleScope software, at which point all changes should be discarded u

IPSetPrefs — If the user chooses to save preferences changes, the TeleScope software will call each I-Piece whose dialog was displayed, and pass in the changed preferences information, after which it will save the data to disk. At this point, the I-Piece may modify its internal preferences to be consistent with the ones passed in.

Copying/Moving Files

Just before the TeleScope Viewer copies or moves files to a new directory, or just before the TeleScope Engine moves files into its "•Processed" directory, the TeleScope sofware calls the I-Pieces to inform them that the files are about to be moved. For each file, the software loads the file's information into an *IPMoveInfo* structure, and calls each I-Piece's *IPMoveFile* routine. The I-Piece may add, remove or change files in the *IPMoveInfo* structure, to instruct the TeleScope software to copy or move different files.

Shut Down

Before the Engine or Viewer software shuts down, it call the I-Piece one last time, with the *IPClose* routine, allowing it to clean up its internal storage.

4. TeleScope Runtime Support

Graphics Support

The TeleScope software contains advanced graphics primitives, which will read and process graphical data in the following formats: TIFF, PCX, JPEG, Targa, BMP, WMF, GIF, EPS, WPG, DIB, PICT and DCX. In the *File Acquisition* case above, an I-Piece may specify that the TeleScope software should handle the file's data as graphics, in which case the graphics primitives will be used to read the file's thumbnail and/or Viewex representation. In the *Viewex Display* case, the I-Piece can specify that the Viewex data should be treated as image data, and the Viewer software will decode and display the image in a window.

The *Standard Image* I-Piece (priority signature 0), which is supplied with the TeleScope software, simply retrieves any default textual information from the file (typically not much), and indicates that the TeleScope software should process the file as image data.

Keyword Support

Also provided with the TeleScope software, is the *Keywords* I-Piece (priority signature 2,147,483,647). This I-Piece, because of its priority value, is guaranteed to be called after all others. Its purpose is to scan through the textual information for a file, and generate keyword values to place into the *keywords* table in the database. Because it is called last, it is unimportant what sort of file is being processed - the *Keywords* I-Piece will always be able to access the text data and generate keyword values.

Global Variables

On the Macintosh, and to a lesser degree on Windows, there are restrictions on the use of global variables in stand-alone code like I-Pieces. To alleviate this situation, the TeleScope software provides a hook whereby I-Pieces can allocate global storage upon starting up, and the calling TeleScope software will assume responsibility for it, passing it back into the I-Piece each time it is called. This is accomplished through the use of the ipStorage parameter described in each of the API calls in the following sections. This parameter is an untyped pointer, which the I-Piece may allocate when it is initialized. Thenceforth, TeleScope will pass this block of storage into the I-Piece at each call, giving it in effect a global store for persistent data. It is the I-Piece's responsibility to dispose of this storage when it is closed.

Preferences

Each I-Piece may be expected to have preference information, usually set by the user, which governs the way it performs its tasks. To relieve the I-Piece of the responsibility for permanently storing its preferences (in a file, for example), TeleScope maintains a block of storage, whose contents are internal to the I-Piece, for preferences information. TeleScope software will be responsible for allocating, disposing, saving and restoring this block. At appropriate times, the TeleScope software will give the I-Piece its preference block, or ask for it for storage.

Additionally, the I-Piece should display a dialog allowing the user to change these preferences when requested by the TeleScope software.

u Important

This arrangement requires that the preferences block be entirely in-line in memory (i.e. no pointers or handles to other structures may be stored in the preferences block). Doing so will not only leak memory within the TeleScope application, but will prevent TeleScope from appropriately saving and restoring your I-Piece's preferences from file. u

Time Sharing

During the File Acquisition and Viewex Display stages, the task that the I-Piece is asked to perform could take considerable time, especially for graphics I-Pieces, which may need to parse large graphics files. During a lengthy operation, the I-Piece can use a call-back procedure pointer provided by the TeleScope software, to share time with the TeleScope software. Included in the I-Piece interface files is a macro, SHARE_TIME, which you can use to automatically call the call-back routine to share processing time. Using the time sharing call-back procedure also allows the TeleScope software to display a progress bar.

Resources (Macintosh)

On the Macintosh, the I-Piece can use resources other than the code resource, located in its resource file. This is useful for localization and minimization of code size. To facilitate this, TeleScope software will ensure that the I-Piece's resource file is the current resource file before calling the I-Piece.

Code Movement (Macintosh)

On the Macintosh, I-Piece code is stored in a code resource, which is loaded into memory in a handle when the calling TeleScope software is started up. In order to safely allow I-Pieces to call Toolbox routines which move or purge memory, the I-Piece's code handle is guaranteed to be loaded and locked before the I-Piece's entry point is called. Thus, you may safely use any Toolbox routine, and may store procedure pointers for the duration of the call.

u Important

After the I-Piece has returned from a call, however, its code resource is unlocked and allowed to float in memory (to prevent fragmentation of the TeleScope software's heap). For this reason, you should not store cross-invocation procedure pointers, as they are not guaranteed to be valid on the next call to the I-Piece. u

5. Application Programming Interface

The following sections describe the calls which are made to I-Pieces by the TeleScope software, in what circumstances they are made, and what data is passed in to them.

Calling Conventions

On the Windows platform, I-Piece files are DLLs, and as such can support dynamic linking and call-by-name routines. Thus, the routine names listed in the sections following are the actual names developers of I-Pieces should use when writing their I-Piece DLL.

For the Macintosh, however, all calls to an I-Piece go through the one entry point at the beginning of the I-Piece's code resource. Thus, the routine names given below are for reference only, and calls will be made with a message parameter, to indicate the desired activity. The definition of the Macintosh code resource entry point is as follows:

```
IPStatus IPEntry (  short ipMessage,
                    Ptr ipParam1,
                    Ptr ipParam2,
                    Ptr ipParam3 );
```

Where *ipParam1*, *ipParam2* and *ipParam3* are generic pointers that pass information into and out of the I-Piece, and *ipMessage* is a parameter that represents the call which is being made by the TeleScope software, in place of calling a named routine as in Windows. Thus, for example, if the windows call to *IPInitialize* is defined as follows:

```
IPStatus IPInitialize (  void** ipStorage );
```

The Macintosh code resource entry point would be called like this:

```
stat = IPEntry(kIPInitialize, &ipStorage, nil, nil);
```

The value of the *ipMessage* parameter indicates which call is being made to the I-Piece, and the *ipParam* parameters are used to pass information to the I-Piece. In the case of *IPInitialize*, *ipParam2* and *ipParam3* are unused and therefore passed as nil. The valid values for *ipMessage* and the parameter values will be detailed with each call.

Return Status

Each I-Piece call must return a value, indicating how the TeleScope software should proceed. All calls to an I-Piece return an *IPStatus* value, which is defined as follows:

```
typedef long int IPStatus;
```

With a few exceptions, noted in the descriptions below, most calls to an I-Piece will return one of the following constants when called:

kStatAbort	-3	The I-Piece encountered some fatal error - do not continue calling other I-Pieces, and do not continue the process which is occurring.
kStatNotHandled	-2	The I-Piece did not handle the call - continue processing by calling the next priority-order I-Piece.
kStatContinue	-1	The I-Piece in some way handled the call - continue processing by calling the next priority-order I-Piece.
kStatHalt	0	The I-Piece completely handled the call - do not continue calling other I-Pieces.

u Important

By returning *kStatHalt* from your I-Piece, you are preventing the TeleScope software from proceeding with normal processing. For example, if your I-Piece returns *kStatHalt* from an *IPAcquireFile* call, the *Keywords* I-Piece will not be called, so you will have to perform your own keyword processing. u

Note that none of the three constants listed above are positive. If an I-Piece returns a positive number as its function result, the calling software will jump directly to the I-Piece with a priority signature greater than or equal to the return value, and continue the calling sequence with that I-Piece. This allows I-Pieces to circumvent the normal processing order, by "skipping" other I-Pieces in the processing chain.

This feature might be used, for example, to prevent other I-Pieces from handling the file, but still allow the *Keywords* I-Piece to perform its processing. To achieve this, an I-Piece would pass back 2,147,483,647 (the priority signature of the *Keywords* I-Piece), causing the TeleScope software to bypass all other I-Pieces and go directly to the the *Keywords* I-Piece.

s Note

If the return value is positive, but less than the I-Piece's own signature value, it is ignored, as though *kStatContinue* had been returned. If an I-Piece cannot be found whose priority signature is greater than or equal to the return value, then I-Piece processing stops normally. s

u Important

Since an I-Piece developer can never be certain which I-Pieces are installed on a user's machine, this feature should be used sparingly. u

IPInitialize

Performs any I-Piece initialization, and allocates the I-Piece's global storage, if any.

```
IPStatus IPInitialize ( void** ipStorage );
```

ipStorage a pointer to a pointer which the I-Piece allocates for its global storage.

IPInitialize is called once only, as soon as the calling TeleScope software has loaded it at startup time. The *ipStorage* parameter initially points to a NIL pointer, which the I-Piece may do with what it pleases. This pointer will be passed in unchanged to all future calls to the I-Piece by the TeleScope software.

On receipt of an IPInitialize call, the I-Piece should allocate any memory it requires during its regular processing (to hold preferences, or other globals), and set the *ipStorage* parameter to point to this memory.

Returns

kStatNotHandled The initialization failed. TeleScope will unload the I-Piece from memory and never call it again.

kStatContinue The initialization was successful.

Macintosh Call Info

ipMessage kIPInitialize

ipParam1 ipStorage (void**)

ipParam2 nil

ipParam3 nil

IPSetPrefs

Provides the I-Piece with a saved preferences block, to govern it's behaviour.

```
IPStatus IPSetPrefs (  void* ipStorage,
                       void* ipPreferences
                       long  ipLength );
```

ipStorage the I-Piece's global storage block (see IPInitialize).

ipPreferences the I-Piece's preferences information, which has been retrieved from a preference file by the TeleScope software. The contents of this block are I-Piece specific.

ipLength the length in bytes of the data pointed to by ipPreferences.

This routine will be called shortly after startup, before any file processing is done by TeleScope, and after the I-Piece preferences dialog is displayed.

On receipt of an IPSetPrefs call, the I-Piece should set its internal structures based on the user preference information contained in the block. The contents of the block are specific to a particular I-Piece, and will have been retrieved from the I-Piece itself

at an earlier time. The I-Piece should make a copy of any information stored in this block, and not store the *ipPreferences* parameter directly, since it will be disposed by TeleScope after the IPSetPrefs call.

Returns

kStatNotHandled	The I-Piece encountered some difficulty setting the preference information. TeleScope will display an alert to this effect, notifying the user, but will continue to use the I-Piece for further processing.
kStatContinue	The I-Piece successfully incorporated the preference information.

Macintosh Call Info

ipMessage	kIPSetPrefs
ipParam1	ipStorage (void*)
ipParam2	ipPreferences (void*)
ipParam3	ipLength (long) - cast from a pointer to a long value

IPAcquireFile

Retrieves textual and graphical information from a specific file.

```
IPStatus IPAcquireFile (  void* ipStorage,
                          IPFileInfo* ipFile );
```

ipStorage	the I-Piece's global storage block (see IPInitialize).
ipFile	an IPFileInfo record describing the file (see elsewhere in this document for a full description of the IPFileInfo record).

When a file is about to be imported into the TeleScope database, this routine will be called for each I-Piece in priority order. The file is open (with read-only priviledge) when the call is made, so the I-Piece may freely read information from the file to populate the IPFileInfo record.

On receipt of an IPAcquireFile call, the I-Piece should use the information in the *ipFile* parameter (as described in Section 6) to read information from the file, and fill in the *ipFile* record as appropriate. See Section 6 for a complete description of the IPFileInfo record's fields, and how the I-Piece should fill them in. On return from this call, the IPFileInfo record will be passed into the next I-Piece in priority sequence, unless the I-Piece indicates otherwise by its return code.

Returns

kStatNotHandled	The I-Piece did not handle the file. The TeleScope software should pass the file to the next I-Piece in priority order.
kStatContinue	The I-Piece changed the information in the IPFileInfo record. The TeleScope software should pass the file to the next I-Piece in priority order.
kStatHalt	The I-Piece changed the information in the IPFileInfo record. The TeleScope software should immediately stop passing the file to other I-Pieces, and process the file into the database.
kStatAbort	The I-Piece determined that the file should not be added to the database. The TeleScope software should ignore this file. Use this return code with care, as TeleScope will not indicate to the user that the file has been ignored.
other	The I-Piece changed the information in the IPFileInfo record. The TeleScope software should immediately jump to the I-Piece whose priority signature is greater than or equal to the return code.

Macintosh Call Info

ipMessage	kIPAcquireFile
ipParam1	ipStorage (void*)
ipParam2	ipFile (IPFileInfo*)
ipParam3	nil

IPClaimDisplay

Determines which I-Piece is capable of handling the viewex display in the Viewer's user interface.

```
IPStatus IPClaimDisplay( void* ipStorage,
                         IPDisplayInfo* ipDisplay );
```

ipStorage	the I-Piece's global storage block (see IPInitialize).
ipDisplay	the IPDisplayInfo record describing the file's information to be displayed (see elsewhere in this document for a full description of the IPDisplayInfo record).

This routine is called when the user initiates the display of the extended view for a particular file. The TeleScope software loads the extended data from the *viewex* table in the database, and calls each I-Piece in priority order, passing in the IPDisplayInfo record.

On receipt of an IPClaimDisplay call, the I-Piece should check the *ipDisplay* parameter to see if it recognizes the extended data (usually the *viewexType* field is used for this purpose, since it was initially set by the I-Piece in the IPAcquireFile call). If it does not, it should immediately return *kStatNotHandled*, so TeleScope can proceed by calling the next I-Piece.

If the I-Piece does recognize the data, there are several options available to it:

1 If the extended data is graphical data, but not in a format supported by the TeleScope software, the I-Piece can read this data into an internal format (usually sharing time as it does so, so TeleScope can display a progress bar), and pass this data back to TeleScope via the *imageData* field in the *ipDisplay* parameter, returning *kStatContinue* from the call. After the IPClaimDisplay call, TeleScope checks this field and if it is not nil, it will use its standard graphical extended view to display the image data to the user. TeleScope will not call the I-Piece further for this extended view (i.e. the IPDrawDisplay, IPDisplayEvent and IPCloseDisplay calls are not made).

2 If the extended data is not graphical data, the I-Piece can take control of the extended view directly. It can set the *windRect* and *windResizable* fields of the *ipDisplay* parameter to appropriate values for the window which will be displayed, and return *kStatContinue*. On return from the call, TeleScope will resize the window to the value in *windRect*, and set the window's resizable characteristic according to *windResizable*, and display the window. TeleScope will set the I-Piece as the "owner" of the extended view window, and will call the I-Piece as appropriate with IPDrawDisplay and IPDisplayEvent calls. When the user closes the extended view window, TeleScope will make a final call to the I-Piece with the IPCloseDisplay call.

3 If the extended data is not a display data type (digital sounds, for example), the I-Piece can request that TeleScope not open a window at all. It can do any processing it requires, and set the *windRect* field of the *ipDisplay* paramter to an empty rectangle (for example, {0, 0, 0, 0}), and return *kStatContinue*. On return from the call, if the *windRect* field contains an empty rectangle, TeleScope will not open a window, but will call the I-Piece's IPDrawDisplay routine exactly once, followed immediately by its IPCloseDisplay routine.

If the extended data is passed to every I-Piece and no I-Piece indicates that they can display the data, TeleScope will attempt to read and display the viewex data as image data, using its internal graphics primitives. Thus, an I-Piece which stores its viewex data in the database as a graphics format recognizable by TeleScope's graphics primitives, does not need to respond to any of the IPDisplay... calls.

Returns

kStatNotHandled	The I-Piece does not recognize this type of Viewex data. TeleScope will proceed with the next I-Piece in priority order.
kStatContinue	The I-Piece has recognized and can display the Viewex data on the screen. Depending on the values of various fields in

the *ipDisplay* parameter, TeleScope will take different actions (see above).

kStatHalt The I-Piece recognized the Viewex data, but an error occurred while processing the data for display. TeleScope will not display an alert, but will close the Viewex window and not call the I-Piece again for this extended view.

IPDrawDisplay

Displays the Viewex information from a specific file in the Viewer's user interface.

```
IPStatus IPDrawDisplay(  void* ipStorage,
                         IPDisplayInfo* ipDisplay );
```

ipStorage the I-Piece's global storage block (see IPInitialize).

ipDisplay the IPDisplayInfo record describing the file's information to be displayed (see Section 6 for a full description of the IPDisplayInfo record).

This routine is called when the Viewex display needs to be drawn (or redrawn) on the screen. This call will only be made to an I-Piece if it had previously returned *kStatContinue* to an IPClaimDisplay call for the same viewex window.

On receipt of an IPDrawDisplay call, the I-Piece should refresh the extended view on the screen. The *viewexWindow* parameter points to the window that needs to be refreshed. TeleScope does not guarantee that this window will be the current graphics port (on the Macintosh) when the call is made.

Returns

The return code from IPDrawDisplay is ignored by TeleScope.

Macintosh Call Info

ipMessage kIPDrawDisplay

ipParam1 ipStorage (void*)

ipParam2 ipDisplay (IPDisplayInfo*)

ipParam3 nil

IPDisplayEvent

Handles a user-generated event in the Viewex view.

```
IPStatus IPDisplayEvent(  void* ipStorage,
                          IPDisplayInfo* ipDisplay );
```

ipStorage	the I-Piece's global storage block (see IPInitialize).
ipDisplay	the IPDisplayInfo record describing the file's information to be displayed (see elsewhere in this document for a full description of the IPDisplayInfo record).

This routine is called when the user generates an interface event (i.e. mouse or key) directed to the Viewex view. Information about the event is contained in the IPDisplayInfo record, and the I-Piece may respond to the event in whatever way is appropriate for the data displayed in the Viewex view.

Returns

kStatNotHandled	The I-Piece did not handle the event. TeleScope will attempt to process the event itself. TeleScope's default processing for events directed at the Viewex view is to do nothing (although standard interface events, such as closing the viewex window, are handled by the TeleScope software).
kStatContinue	The I-Piece has handled the event, and the Viewex window should be left on the screen.
kStatHalt	The I-Piece has handled the event, and the Viewex window should immediately be closed.

Macintosh Call Info

ipMessage	kIPDisplayEvent
ipParam1	ipStorage (void*)
ipParam2	ipDisplay (IPDisplayInfo*)
ipParam3	nil

IPCloseDisplay

Gives the I-Piece an opportunity to clean up before the Viewex view is closed.

```
IPStatus IPCloseDisplay(  void* ipStorage,
                          IPDisplayInfo* ipDisplay );
```

ipStorage	the I-Piece's global storage block (see IPInitialize).
ipDisplay	the IPDisplayInfo record describing the file's information to be displayed (see elsewhere in this document for a full description of the IPDisplayInfo record).

This routine is called just before the Viewex view is removed from the screen, and its window closed, either in response to a user action (such as clicking in the window's

close box), or in response to a result code of *kStatHalt* from an IPDisplayEvent call. The I-Piece may clean up any storage it has allocated for the display.

Returns

Regardless of the return code from IPCloseDisplay, the Viewex view will be removed from the screen.

Macintosh Call Info

ipMessage	kIPCloseDisplay
ipParam1	ipStorage (void*)
ipParam2	ipDisplay (IPDisplayInfo*)
ipParam3	nil

IPDisplayPrefs

Displays the I-Piece's preferences dialog, and allows the user to change its values.

```
IPStatus IPDisplayPrefs(   void* ipStorage,
                           void* ipPreferences
                           long ipLength );
```

ipStorage	the I-Piece's global storage block (see IPInitialize).
ipPreferences	the I-Piece's preference information, which contains the current information to display in the preferences dialog, and which will be changed by the I-Piece to reflect user changes in the dialog. The contents of this block are I-Piece specific.
ipLength	a long integer whose value is the length in bytes of the data pointed to by ipPreferences.

This routine is called from both the Engine and Viewer software, when the user chooses the "I-Piece Preferences..." menu item. A window containing icons for each I-Piece is displayed, and when the user clicks on an icon, the appropriate I-Piece's IPDisplayPrefs routine is called.

On receipt of an IPDisplayPrefs call, the I-Piece should display a dialog (which must be an application modal dialog) containing the preferences settings, and allow the user to change these settings before returning. On return, if the result code of the IPDisplayPrefs function call is *kStatContinue*, TeleScope will take the changed preferences information from *ipPreferences*, save it to the preferences file, and immediately call the I-Piece's IPSetPrefs routine, to make the changes permanent.

Returns

kStatContinue	The user changed values in the settings dialog, and approved the changes by clicking in the OK button. The changed preference information should be saved.
kStatHalt	The user dismissed the settings dialog with the Cancel button. The preference information should be discarded.

Macintosh Call Info

ipMessage	kIPDisplayPrefs
ipParam1	ipStorage (void*)
ipParam2	ipPreferences (void*)
ipParam3	ipLength (long) - cast from a pointer to a long value

IPGetPrefs

Retrieves a copy of the current preferences from the I-Piece.

```
IPStatus IPGetPrefs (  void* ipStorage,
                       void** ipPreferences,
                       long* ipLength );
```

ipStorage	the I-Piece's global storage block (see IPInitialize).
ipPreferences	a pointer to a pointer which the I-Piece will allocate for a copy of its current preference settings. Since TeleScope will save this information to a preferences file, and then delete it, it must be a copy.
ipLength	a pointer to a long integer, which the I-Piece will fill with the length, in bytes, of the data pointed to by ipPreferences.

TeleScope calls this routine just before calling IPDisplayPrefs when displaying the I-Piece's preferences dialog, and just before calling IPClose when the TeleScope software shuts down. The *ipPreferences* parameter initially points to a nil pointer, which the I-Piece must allocate. Once this call returns, TeleScope takes responsibility for the allocated block, and will deallocate it after it has finished with it.

s Macintosh Note

Because the TeleScope software on the Macintosh uses DisposePtr to deallocate the *ipPreferences* block, the I-Piece must allocate the block using NewPtr, not malloc. s

Returns

kStatContinue	Success. The copy of the preferences has been allocated in *ipPreferences*.
kStatHalt	Failure. The I-Piece should deallocate any memory it has created in the process of attempting to generate the copy of the preferences, and set *ipPreferences* to point to a nil pointer.

Macintosh Call Info

ipMessage	kIPGetPrefs
ipParam1	ipStorage (void*)
ipParam2	ipPreferences (void**)
ipParam3	ipLength (long*)

IPMoveFile

Allows the I-Piece to add, modify or remove files to be copied or moved by the TeleScope software.

```
IPStatus IPMoveFile (  void* ipStorage,
                       IPMoveInfo* ipInfo );
```

This routine is called when the TeleScope software is about to move the original file for a managed document to a new location in the file system. The TeleScope software will resolve the location of the original file, mounting shared or removable volumes if necessary, and fill in the *ipInfo* record with information about the file to be moved. It then calls the IPMoveFile routine for each I-Piece in priority sequence, passing this information in. The I-Piece may add, remove, or change files in the *ipInfo* record before returning, and the TeleScope software will move all the files pointed to by *ipInfo* to the new location.

Returns

kStatNotHandled	The I-Piece did not handle the call. The TeleScope software should proceed to the next I-Piece in priority order.
kStatContinue	The I-Piece changed the information in the IPMoveInfo record. The TeleScope software should proceed to the next I-Piece in priority order.
kStatHalt	The I-Piece changed the information in the IPMoveInfo record. The TeleScope software should stop calling I-Pieces, and immediately move the files in the IPMoveInfo record.
kStatAbort	The I-Piece determined that the file in the IPMoveInfo record should not be moved. The TeleScope software will not move

	the file to the new location. Use this return code with caution, as the TeleScope software does not notify the user that the file was not moved.
other	The I-Piece changed the information in the IPMoveInfo record. The TeleScope software should immediately jump to the I-Piece whose priority signature is greater than or equal to the return code, and continue processing.

Macintosh Call Info

ipMessage	kIPMoveFile
ipParam1	ipStorage (void*)
ipParam2	ipInfo (IPMoveInfo*)
ipParam3	nil

IPClose

Terminates the I-Piece.

```
IPStatus IPClose (  void* ipStorage );
```

ipStorage	the I-Piece's global storage block (see IPInitialize).

This routine is called immediately before the TeleScope software shuts down. It is the I-Piece's responsibility to dispose of its global storage in *ipStorage*.

Returns

Regardless of the return code, TeleScope will continue to shut down.

Macintosh Call Info

ipMessage	kIPClose
ipParam1	ipStorage (void*)
ipParam2	nil
ipParam3	nil

6. Data Structures

There are three main data structures which are used by the I-Piece API. The IPFileInfo record is used when adding a file to the database, the IPDisplayInfo record is used when displaying a file's Viewex data on the screen within the Viewer interface, and the IPMoveInfo record is used when copying or moving a file to a new location in the file system.

IPFileInfo Record

The IPFileInfo record describes all the information that will be inserted into the database for a given file, and a great deal of file information besides. This record is used by the I-Piece to determine whether or not it wishes to handle the file, and if so, to act as a repository for data extracted from the file. TeleScope will then use this extracted data for insertion into the database. The IPFileInfo record is defined as follows:

```
struct IPFileInfo {
     FILE_HDL openFile;
     FILE_REF fileOnDisk;
     HENV odbcEnv;
     HDBC odbcConn;
     SHARE_PTR shareTime;

     char fileType[4];
     char fileCreator[4];
     char fileName[64];
     unsigned long fileSize;
     short fileVersion;
     LONG_DT fileDate;
     long paperClip;
     char category[10];
     char subCategories[128];
     char country[32];
     char state[32];
     char city[32];
     char shortDescription[256];
     TEXT_REF longDescription;
     char userNotes[256];
     char transName[32];
     char transService[32];
     char creator[256];

     TEXT_REF keywords;

     unsigned long thumbOffset;
     IMAGE_REF thumbnail;

     unsigned long viewexOffset;
     char viewexType[4];
     BINARY_REF viewex;

     char fileInfo[256];
};
```

The values of many of the fields in this record are sent directly to the *editorial* table in the TeleScope database. For a complete description of the contents of these fields, see the document *TeleScope Design Documentation: Database Design*.

Field	Type	Use	Description
openFile	FILE_HDL	in	This is a platform-specific reference to the file, which has already been opened (with read-only privilege) by TeleScope. This file handle can be used to read from the file. On the Macintosh, FILE_HDL is a short integer, which refers to the open data fork of the file. On Windows, FILE_HDL is a void*.
fileOnDisk	FILE_REF	in	This is a platform-specific reference to the file on disk. You may need this to gain access to the file's name, or open the resource fork of the file (on the Mac). On the Macintosh, FILE_REF is an FSSpec record. On Windows, FILE_REF is a TCHAR*.
odbcEnv	HNEV	in	This is an ODBC-specific reference to the current ODBC environment. The I-Piece may use this environment, and its associated connection shown below, to contact the database into which the file will be inserted.
odbcConn	HDBC	in	This is an ODBC-specific reference to the current ODBC connection. The I-Piece may use this, along with the *odbcEnv* field, to communicate with the database into which the file will be inserted.
shareTime	SHARE_PTR	in	This is a pointer to a routine which the I-Piece should call during lengthy processing while dealing with the file. This routine should be called at least once a second to ensure that other processes running on the computer have a chance to proceed. You can use the SHARE_TIME macro provided in the I-Piece interfaces to share time using this pointer.
fileType	char[4]	in	This is the 4-character type of the file, as defined on the Macintosh. For Windows files, this field should be filled in by the Engine or Viewer software before calling

			the I-Piece, perhaps based on the file's extension. All 4 characters of this field must be used - pad with spaces if necessary.
fileCreator	char[4]	in	This is the 4-character creator signature of the file, as defined on the Macintosh. For Windows files, this field should be left blank. All 4 characters of this field must be used - pad with spaces if necessary.
fileName	char[64]	in	This is the name of the file, without any preceding path specification. The text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 63.
fileSize	unsigned long	in	This is the size of the file in bytes on disk. The I-Piece should not modify this value.
fileVersion	short	out	The I-Piece should set the version number of the file. Initially, this field is set to zero.
fileDate	LONG_DT	in/out	On input, this field is set to the creation date of the file. If the I-Piece wishes to change this information, it may do so. On the Macintosh, LONG_DT is defined as an unsigned long value. On Windows, LONG_DT is defined as an unsigned long.
paperClip	long	out	The I-Piece should set the paper clip value for this file. The paper clip is used to tie several related files together in the database.
category	char[10]	out	The I-Piece should fill this field in with textual information from the file. The text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 9.
subCategories	char[128]	out	The I-Piece should fill this field in with textual information from the file. The text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 127.

country	char[32]	out	The I-Piece should fill this field in with textual information from the file. The text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 31.
state	char[32]	out	The I-Piece should fill this field in with textual information from the file. The text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 31.
city	char[32]	out	The I-Piece should fill this field in with textual information from the file. The text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 31.
shortDescription	char[256]	out	The I-Piece should fill this field in with textual information from the file. The text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 255.
longDescription	TEXT_REF	out	The I-Piece should fill this field in with textual information from the file. On the Macintosh, TEXT_REF is defined as a Handle to the text data, which is initially passed in nil. The I-Piece must allocate the Handle using NewHandle if it is going to be storing data in it. On Windows, TEXT_REF is defined as TCHAR* which is initially passed in nil, and must be null terminated when returned. The I-Piece must allocate a block of memory using malloc if it is going to be storing data in it.
userNotes	char[256]	out	The I-Piece should fill this field in with textual information from the file. The text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 255.
transName	char[32]	out	The I-Piece should fill this field in with textual information from the file. The

			text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 31.
transService	char[32]	out	The I-Piece should fill this field in with textual information from the file. The text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 31.
creator	char[256]	out	The I-Piece should fill this field in with textual information from the file. The text in this field should be a null-terminated (C-style) string, limiting the effective number of characters in this field to 255.
keywords	TEXT_REF	out	On Macintosh, TEXT_REF is defined as a Handle to textual information, which is initially passed in nil. The I-Piece must allocate space for it using NewHandle if it is going to store information in it. On Windows, TEXT_REF is defined as TCHAR* which is initially passed in nil, and must be null-terminated when returned. The I-Piece must allocate a block of memory using malloc if it is going to be storing data in it. The TEXT_REF will contain any number of words, separated by spaces, which will be used in the *keywords* table as references to this file.
thumbOffset	unsigned long	out	This field and the *thumbnail* field, are used to specify the thumbnail information for the TeleScope software. If *thumbnail* is nil, then *thumbOffset* will be assumed to be an offset into the file where TeleScope's image processing engine can find image data to process into a thumbnail. If *thumbnail* is non-nil, then *thumbOffset* is ignored (and should be set to zero).
thumbnail	IMAGE_REF	out	This is a platform-specific representation for the thumbnail image data, if it cannot be extracted from the file by TeleScope's graphic processing engine. On the Macintosh, IMAGE_REF is defined as a

			PicHandle, which returns a Macinsoth picture to TeleScope, which will convert that into the JPEG thumbnail representation in the database. On Windows, IMAGE_REF is defined as an HBITMAP which returns a DIB that TeleScope will convert into the JPEG thumbnail representation in the database.
viewexOffset	long integer	out	This field and the *viewex* field, are used to specify the viewex information for the TeleScope software. If *viewex* is nil, then *viewexOffset* will be assumed to be an offset into the file where TeleScope's image processing engine can find image data to process into a JPEG viewex. If *viewex* is non-nil, then *viewexOffset* is ignored (and should be set to zero).
viewexType	char[4]	out	A 4-character code representing the type of data stored in the *viewex* field. All 4 characters of this field must be used - pad with spaces if necessary. If the I-Piece is allowing TeleScope's image processing engine to handle the viewex data (i.e. if the *viewex* field is nil), then this field is ignored by TeleScope.

viewex	BINARY_REF	out	This is a platform-specific representation for the viewex data, if it cannot be extracted from the file by TeleScope's graphic processing engine. On the Macintosh, BINARY_REF is defined as a Handle to binary data, which TeleScope will store directly in the database. On Windows, BINARY_REF is defined as a structure which has a length and a pointer to the binary data, as follows:
			`typedef struct {`
			`  DWORD  dwLength;`
			`  BYTE*  pbData;`
			`} BINARY_REF;`
			The pbData field must be allocated using malloc, and will be freed automatically by TeleScope.

file_info	char[32]	out	This is a field which is stored in the database and shown to the user when they view the file in the editorial view of TeleScope Viewer. This field can contain

any information that the I-Piece wishes to display to the user about the file. If the TeleScope software is asked to read the file (i.e. if *thumbnail* or *viewex* are nil), and the I-Piece has left the file_info field blank, the TeleScope software will add default image information to the file_info field, as follows: "*x* by *y* pixels, *a* pixels/inch, *b* bits/pixel"

s Macintosh Note

On the Macintosh, there are cases when you may wish to use a Picture handle to specify viewex data as well (such as for EPS files, where the placement image is actually a full-sized PICT resource). TeleScope provides two special features which allow I-Pieces to do this:

1) If the *viewexType* field is set to 'PICT', and a PicHandle is stored in the *viewex* field, then TeleScope will treat the handle as picture data, and convert it to JPEG automatically.

2) The *viewex* and *thumbnail* fields can both be set to be handles to the same picture data. In this case, TeleScope will use the same picture data for both thumbnail (scaled down appropriately) and viewex images and, more importantly, correctly dispose of the picture handle only once, avoiding a possible system crash.

These features also mean that you must be careful when disposing of an existing *thumbnail* or *viewex* handle, however. Before disposing either handle, ensure that the *thumbnail* and *viewex* handles do not point to the same memory. If they do, do not dispose the handle, or dispose of it only once! s

s Macintosh Note

Any handles in the IPFileInfo record (i.e. TEXT_REF, IMAGE_REF, or BINARY_REF) are disposed of automatically by the TeleScope software when they are no longer needed, using DisposeHandle. This means that if the I-Piece is retrieving any of these handles from a resource using GetResource (or GetPicture), the I-Piece must also call DetachResource before returning the handles. s

s Windows Note

Any pointers in the IPFileInfo record (i.e. TEXT_REF, IMAGE_REF, or BINARY_REF) are disposed of automatically by the TeleScope software when they are no longer needed, using free. Handles in the IPFileInfo record (i.e. IMAGE_REF) will be disposed of using CloseHandle. s

u Important

The I-Piece will always be called after at least one other I-Piece (the Graphics I-Piece is always called first). This means that other I-Pieces may have added data to the

IPFileInfo record before you get called. Before setting data into a pointer or handle field, either make sure that any existing data in that field has been appropriately disposed (as described above), or append your data to the existing pointer or handle. Failure to do so will cause TeleScope to leak memory and could eventually cause an out-of-memory condition to occur (especially during large imports). u

IPDisplayInfo Record

This record is used by the IPDrawDisplay, IPDisplayEvent, and IPCloseDisplay calls, to give the I-Piece any required information about the viewex view in which the Viewex data is being displayed. The IPDisplayInfo record is defined as follows:

```
struct IPDisplayInfo {
    char viewexType[4];
    BINARY_REF viewexData;
    ALIAS_REF viewexFile;
    WIND_REF viewexWindow;
    RECT_REF windRect;
    EVENT_REF windEvent;
    void* ipRefCon;
    BOOL_REF windResizable;
    GWORLD_REF imageData;
    SHARE_PTR shareTime;
};
```

Field	Type	Use	Description
viewexType	char[4]	in	The 4-character type indicating what sort of data is stored in the *viewexData* field. This information was created by the I-Piece originally, in the IPFileInfo record above.

viewexData	BINARY_REF	in	A platform-specific reference to the binary viewex data. On Macintosh, BINARY_REF is defined as a Handle to binary data. On Windows, BINARY_REF is defined as a structure which has a length and a pointer to the binary data, as follows:
			`typedef struct {`
			`  DWORD dwLength;`
			`  BYTE* pbData;`
			`} BINARY_REF;`
			The pbData field is allocated using malloc, and will be freed automatically by TeleScope.

viewexFile	ALIAS_REF	in	A platform-specific reference to the original file's location on disk. Certain I-Pieces may require access to the original

			file to display an extended view or preview of the file. Rather than requiring the I-Piece to redundantly store a reference to the original file in the viewex data itself, TeleScope provides this field for all files, which the I-Piece may ignore if it does not require it. On Macintosh, ALIAS_REF is defined as an AliasHandle which can be resolved to an FSSpec record using the Toolbox call ResolveAlias. On Windows, ALIAS_REF is defined as a TCHAR* which points to a null-terminated string which is in the same format as the file_location field in the TeleScope Database (see document *TeleScope Design Documentation - Cross-Platform File Specifications*)
viewexWindow	WIND_REF	in	A reference to the window in which the viewex data will be drawn. TeleScope creates this window initially, and is responsible for disposing of it when it is closed. All viewex drawing should be done inside this window. On Macintosh, WIND_REF is defined as a WindowRef. On Windows, WIND_REF is defined as an HDC to a device context.
windRect	RECT_REF	in/out	On input, this field contains the window's rectangle, in screen coordinates. After the *IPClaimDisplay* call, TeleScope checks this field and, if its contents have changed, will resize the Viewex window appropriately. On Macintosh, RECT_REF is defined as a Rect. On Windows, RECT_REF is defined as a RECT. The Viewer software will not resize the window smaller than a user-defined minimum, or larger than the size of the current screen. Values returned by the I-Piece in *windRect* will be corrected within these boundaries after the I-Piece is called. s Note If, on the IPClaimDisplay call, the I-Piece sets this rectangle to an empty rectangle, the Viewer software will not open a window for the viewex data. In this case, the Viewer software will call the I-Piece's

			IPDrawDisplay routine once, immediately followed by its *IPCloseDisplay* routine. s

windEvent	EVENT_REF	in	A platform-specific event descriptor, which details the user event (i.e. mouse or key) which was directed at the Viewex window (and therefore must be handled by the I-Piece). This field is only valid during the IPDisplayEvent call - all other times it is nil. On Macintosh, EVENT_REF is defined as an EventPtr. On Windows, EVENT_REF is defined as a struct containing the Windows event message, and the wParam and lParam fields, as follows:
			`typedef struct {`
			`  UINT  uMessage;`
			`  WPARAM  wParam;`
			`  LPARAM  lParam;`
			`} EVENT_REF;`

ipRefCon	void*	in/out	An untyped pointer, which is initially passed in nil to the *IPClaimDisplay* call, and is not altered by the TeleScope software after that point. This field can be used by the I-Piece for whatever it requires while the display is up. At the *IPClaimDisplay* call, TeleScope sets the field to nil before the call, and the I-Piece may allocate a block of storage, setting *ipRefCon* to point to it. On all subsequence *IPDrawDisplay* or *IPDisplayEvent* calls for that extended view, TeleScope leaves the *ipRefCon* field untouched. The I-Piece is responsible for disposing of any memory it has allocated in the *ipRefCon* field. It should do this during the *IPCloseDisplay* call.
windResizable	BOOL_REF	out	During the *IPClaimDisplay* call, the I-Piece can set this field to TRUE or FALSE, indicating whether the extended view window (*viewexWindow*) should be resizable by the user or not. For all other calls, the value in this field is ignored.
imageData	GWORLD_REF	out	A GWORLD_REF is defined as a GWorldPtr on the Macintosh, and an HBITMAP handle under Windows. Before the *IPClaimDisplay* call, TeleScope

			sets this field to nil. During the *IPClaimDisplay* call, the I-Piece may set this field to point to some graphics representing a preview for the file, and pass it back. TeleScope will display the data in its standard graphics extended view window, and will not call the I-Piece again for the display (i.e. no *IPDrawDisplay*, *IPDisplayEvent* or *IPCloseDisplay* calls).
shareTime	SHARE_PTR	in	During the *IPClaimDisplay* call, the I-Piece can share time and display a progress bar while it is loading the extended view data or preparing it for display.

IPMoveInfo Record

When the TeleScope software needs to move a managed document from one location to another within the file system, it first locates the original file, then loads up the *IPMoveInfo* record with information from the database about the file, and calls the *IPMoveFile* routine in each I-Piece in priority order. Any I-Piece which recognizes the file information can make changes to the *IPMoveInfo* record; specifically, the I-Piece can add files to the list of files to be moved, and the TeleScope software will copy or move all the files together.

```
struct IPMoveInfo {
    char fileType[4];
    short numFiles;
    FILE_REFPTR files;
};
```

Field	Type	Use	Description
fileType	char[4]	in	The 4-character file type of the document being moved, taken directly from the *editorial* table in the database (not from the file itself).
numFiles	short	in/out	On input, the number of files pointed to in the *files* array (which will always initially be 1). On output, the I-Piece sets this to the number of files it has placed into the *files* array.
files	FILE_REFPTR	in/out	On input, this field points to an array of one (1) FILE_REF structure (see below). The I-Piece, if it is adding files to the array, must first allocate space for the new array (using realloc on Windows, or SetPtrSize on the Macintosh), and set *files*

to point to the new array. On the Macintosh, FILE_REF is an FSSpec record. On Windows, FILE_REF is a TCHAR[260] array which contains a path to the file.

Appendix A: Developer Support

North Plains Systems Inc. is committed to supporting third-party developers who develop I-Pieces for TeleScope. This document is only one example of this. The following are some of the developer services offered by North Plains Systems:

- Developer Pak: A complete development package, consisting of a number of sample I-Pieces and electronic documentation, will help you get started, and provide you with a framework in which to build your I-Piece.
- E-Mail Tech Support: We provide developer support via e-mail - simply mail your questions to the e-mail address provided below.
- FTP Access: To access our technical documentation on-line, use the FTP site listed at the address below. Technical notes are updated often, and product upgrades are also available.
- Web Tech Support: At the Web address shown below, using a World-Wide Web browser such as Mosaic or NetScape, you can access our on-line documentation and technical support via HTML.
- Telephone Support: Although you will be guaranteed a quicker response time using our e-mail technical support, in an emergency, we may be able to help you over the phone. Our main contact number is listed below.

North Plains Systems Inc.
4094 Tea Garden Circle
Mississauga, Ontario
CANADA L5B 2X8

phone: (905) 272-9186
fax: (905) 272-0413
e-mail: devsupport@northplains.com
ftp: ftp.northplains.com/pub/devsupport
www: www.northplains.com/devsupport.html