



April 29-30, 2003 Meeting Agenda

- ▶ [Home](#)
- ▶ [Documents](#)
- ▶ [Resources](#)
- ▶ [Mailing Lists](#)
 - [RETS-L](#)
 - [RETS-Announce](#)
 - [RETS-Dev](#)
- ▶ [Working Groups](#)
 - [Compliance](#)
 - [Data Dictionary](#)
 - [Metadata](#)
 - [Transport](#)
 - [Update](#)
- ▶ [Comments](#)
- ▶ [Links](#)

April 29-30, 2003

	Time	Discussion Item
		<u>April 29</u>
	9:30 - 12:00	Compliance Workgroup Meeting
	12:00 - 1:00	Lunch
	1:00 - 1:10	Introductions
	1:10 - 1:30	Update on other standards groups
	1:30 - 1:45	Release and meeting timing
	1:45 - 2:30	Compliance Discussion (this may be omitted if substantially all attendees also participate in the compliance workgroup discussions in the morning).
	2:30 - 2:45	Break
	2:45 - 3:30	Group discussion issue: dynamic metadata updates
	3:30 - 4:30	Change proposal discussion
		<u>April 30</u>
	8:00 - 8:30	Breakfast
	8:30 - 10:15	Change proposal discussion
	10:15 - 10:30	Break
	10:30 - 12:00	Change proposal discussion
	12:00 - 1:00	Lunch and wrap-up

Dress code for the meeting is Business Casual

Last changed 21 April 2003 by [Webmaster](#).

RETS Change Proposal 30: Hotsheet

Author: Maggi Diaz

Organization: WyldFyre

Address: 900 East Hamilton Ave., Suite 500, Campbell, CA 95008

Telephone Number: (408) 369-8900

Email: maggie@wyldfyre.com

Status: Proposal

Date: January 20, 2003

Version: 1.5

1. Synopsis

This change proposal is to (1) add Hotsheet as another Resource in the metadata structure, (2) add Hotsheet as another Search Type, and (3) add Hotsheet to the RETS DTD.

2. Rationale

RETS vendors need a standard to implement Hotsheet that is currently not in RETS

3. Proposal

3a. Include Hotsheet as another Search Type in Chapter 7, Section 7.1 Search Types, and describe as follows:

Hotsheet A Hotsheet Search is a search against a property class database/table.

3b. Include Hotsheet as another Resource in Section 11.2.2. Resource, Table 11-1 Well-Known Resource Names as follows:

Resource	Name	Purpose
Hotsheet		A resource that contains information about property listings with price and status changes from a resource property class table.

3c. Include Hotsheet in Real Estate Data (REData) Transaction DTD, Chapter 1, Section 1.4.8 Packaging Elements, as follows:

```
REData
REProperties?
  ResidentialProperty*
  CommonInterest*
  LotsAndLand*
  MultiFamily*
  TaxData*
REOffices?
  REOffice+
REAgents?
  REAgent+
REOfficeRosters?
  REOfficeRoster+
REActivities?
  Activity+
REProspects?
  Prospect+
REPublicRecords?
  RETax+
REHistories?
  REPropHistory+
  REPropEntry+
REHotsheet?
  REHotsheetRecord+
```

3d. Include Hotsheet in Real Estate Data (REData) Transaction DTD, Section 1.5 Comments on Field Usage, as follows:

The ListingID is the intended container for the data in the KeyField of the Hotsheet Resource as defined in the Metadata.

3e. Include Hotsheet in Real Estate Data (REData) Transaction DTD, Chapter 2 DTD, as follows:

```
<!-- PACKAGING ELEMENTS -->
<!ELEMENT REHotsheet (REHotsheetRecord+) >
```

3f. Include Hotsheet in the Real Estate Data (REData) Transaction DTD, Chapter 2 DTD, as follows:

Note1: The REHotsheetRecord utilizes the 'PropertyRecord' element defined in the 'Listing History' section. That element should be moved from the 'Listing History' section to the general 'Compound Elements' section.

Note2: The REHotsheetRecord utilizes the 'ChangeType' element defined in the 'Listing History' section. That element should be modified as shown below and moved from the 'Listing History' section to the general 'Compound Elements' section.

```
<!ELEMENT ChangeType (#PCDATA) >
  <!ATTLIST ChangeType (#PCDATA) >
    PriceChange (Increase | Decrease | None)
    StatusChange (Yes | No)
    System (Yes | No) "No" >

<!-- Hotsheet Elements -->
<!ELEMENT PreviousListPrice (#PCDATA) >
<!ATTLIST PreviousListPrice (#PCDATA) >
  Type (INTEGER | FLOAT) "INTEGER"
  CurrencyCode CDATA "USD" >

<!ELEMENT PreviousStatus (#PCDATA) >
<!ATTLIST PreviousStatus (#PCDATA) >
  Status (Active | Closed | Expired | OffMarket | Pending) >

<!-- Compound Hotsheet Elements -->
<!ELEMENT REHotsheetRecord (ListingID, ChangeType, PreviousListPrice?,
PreviousStatus?, PropertyRecord?, ModificationTimestamp?)>
```

3g. Include Hotsheet in Real Estate Data (REData) Transaction DTD, Chapter 3 Sample Data, as follows:

```
3.7 Hotsheet
<REData>
  <REHotsheet>
    <REHotsheetRecord>
      <ListingID>4839424</ListingID>
      <ChangeType PriceChange="Increase" StatusChange="No">PriceIncrease</ChangeType>
      <PreviousListPrice>489000</PreviousListPrice>
      <PreviousStatus Status="Active">New</PreviousStatus>
      <PropertyRecord>
        <ResidentialProperty>
          <Listing>
            <StreetAddress>
              <StreetNumber>720</StreetNumber>
              <StreetDirPrefix>E</StreetDirPrefix>
              <StreetName>Vineyard</StreetName>
              <StreetSuffix>Ln</StreetSuffix>
            </StreetAddress>
            <ListingData>
              <ListPrice>479000</ListPrice>
            </ListingData>
            <SalesData>
              <ClosePrice>0</ClosePrice>
            </SalesData>
            <MLSInformation>
              <ListingStatus Status="Active">Active</ListingStatus>
              <StatusChangeDate>2003-01-31T8:00:00</StatusChangeDate>
              <ModificationTimestamp>2003-01-31T15:25:40.897</ModificationTimestamp>
            </Listing>
          </ResidentialProperty>
        </PropertyRecord>
      <ModificationTimestamp>2003-01T15:25:40.897</ModificationTimestamp>
    </REHotsheetRecord>
  </REHotsheet>
</REData>
```

4. Development Impact

This proposal adds another Resource and Search Type in RETS. There should not be an impact on RETS clients and servers.

This proposal is an enhancement on the specification and should not impact current or future RETS client and server implementations.

5. Compatibility

None

5. Proof/Need of Concept Examples

None

Last changed 13 February, 2003 by [Webmaster](#).

RETS Change Proposal 31: Update Warning Block Enhancement

Author: Sergio Del Rio

Organization: Templates 4 Business, Inc.

Telephone Number: (604) 529-1544

Email: Sergio.Del.Rio@t4bi.com

Status: Proposal

Date: March 24, 2003

Version: 1.0

1. Synopsis

This change request will enhance the server's ability to maintain an audit trail of non-fatal rule violations by enhancing the Update Transaction. This enhancement will allow the server to indicate which warnings require end-user feedback and will allow a client to provide this feedback to the server.

2. Rationale

Many servers that implement the Update Transaction require support for warnings in addition to errors. Warnings are generally used to indicate that a rule violation has occurred that was not fatal and did not prevent the record from being saved into the database. While the current approach is sufficient for many implementations, most specifically a batch update program, it does not allow for a warning to be acknowledged by an interactive end-user in any way.

In some environments, it is required that an end-user provide a valid reason for having violated certain warnings.

To support this common business situation, it is proposed that a warning block be added to the Update Response Body Format that will give the server a clear way of indicating warnings separately from errors. Additionally, a new request argument is proposed for the Optional Request Arguments section. The new request argument will be called WarningResponse. It will allow a response to be provided by an end-user.

3. Proposal

This proposal affects several sections of the Update Transaction as follows.

3.1 Add warning-block to the Update Response Body Format

The warning-block is designed to allow the server to send back warnings (errors which will not prevent a record from being saved to the database). For consistency, it will be created identically to the error-block with the addition of a single field.

The warning-block is defined as follows:

```
warning-block ::= <WARNINGBLOCK>CRLF
                1*(<WARNINGDATA>field<delimiter>warning-
                num<delimiter>warning-offset<delimiter>warning-text<delimiter>response-
                required</WARNINGDATA>CRLF)
                </WARNINGBLOCK>CRLF
```

A Warning Block is returned when there is a problem with one or more of the fields that did not prevent the record from being saved to the database. It contains the field name, a warning number, some additional text about the warning (warning-text), where in the field data the warning occurred (warning-offset) and an indicator whether an end-user response to this warning is requested or required. The delimiter will be the same delimiter as the one defined for the error-block.

```
field ::= SQLFIELDNAME
```

This is the field name that generated the warning.

```
warning-num ::= 1*5DIGIT
```

This is the host warning number. This number, along with the warning-text, MAY be displayed to an end-user when looking at the corresponding field in the client application.

```
warning-offset ::= 1*5DIGIT
```

This is the offset into the field data that was sent by the client application to the server. It indicates at what character in the field data the problem was encountered. This number is set to zero ("0") if the offset of the error is unknown.

```
error-text ::= *256ALPHANUM
```

This is the warning text generated by the host to assist an end-user in determining the problem with the field data. This text is associated with the warning-num.

```
response-required ::= 0 | 1 | 2
```

This indicates whether an end- user response to this warning is requested or required as follows:

- 0 - No response is requested
- 1 - A response is requested
- 2 - A response is mandatory

If the response is mandatory, the client MUST send the end-user response for the specific warning-num in the WarningResponse request argument in order for this record to be successfully saved to the database.

3.2 Add WarningResponse to the Optional Request Arguments

This is a key-value pair indicating the responses which were provided for each warning-num that had the response-required field set to 1 or 2.

The key-value MUST be formatted as follows:

```
warning-num = warning-response *(field-delimiter warning-num = warning-response)
```

The definitions of the key-value pair fields are:

```
warning-num ::= 1*5DIGIT
```

This is the host warning number that was returned in the Update Response Body Format.

```
user-response ::= *256ALPHANUM
```

This is the text that the end-user entered in response to the specified warning. If a warning-num has been indicated as a response required of 2, then this string MUST NOT be null.

3.3 Additional Reply Codes

20211 WarningResponse was not given for all warnings that indicated a response-required value of 2.

4. Development Impact

Development impact depends on each current RETS client and server implementations, but should not have any impact on future RETS client and server implementations. Each existing RETS server and client, prior to passing of this proposal, have the option to change accordingly to be RETS compliant. Future RETS server and client implementation after approval of RETS 1.5 MUST follow RETS 1.5 standard.

5. Compatibility

Servers that do not send back any warning-block will remain backwards compatible to the existing RETS 1.0 specification and can continue to function and be considered RETS 1.5 compatible. It is not mandatory for a server to send back a warning-block.

5. Proof/Need of Concept Examples

None

Last changed 08 April, 2003 by [Webmaster](#).

RETS Change Proposal 32: Distributed Data Base Enhancements

Author: Leo Bijngte

Organization: Fidelity National Information Systems

Telephone Number: (408) 369 - 8900

Email: leob@fnis.com

Status: Proposal

Date: March 31, 2003

Version: 1.0

1. Synopsis

This proposal adds functionality to RETS to permit clients to maintain local copies of a data base efficiently and accurately with minimal impact on the server. It includes changes to metadata, an enhancement to the Search transaction to facilitate returning data in manageable chunks, and a new facility for querying server management parameters.

2. Rationale

RETS currently lacks some key elements necessary to perform a common function: replication of selected parts of a server's data. This is currently possible in RETS using tools such as modification timestamps, the search transaction `Limit` and `Offset` parameters, `METADATA-FOREIGN_KEYS` and so on. However, replication using these tools is imperfect and often requires special arrangements between endpoints. This is contrary to the goal of RETS, which is to obviate the need for such special point-to-point arrangements.

The proposal requires changes to the standard in three areas. First, the RETS mechanism for dealing with limits on the number of returned records is modified so that it's possible to retrieve data sets larger than the general search site limit without causing a significant server load. This is done by putting the server in control of the chunking mechanism and chunked transfers. There is also a mechanism that allows a client to query a server as to the minimum permissible time between updates, and the server is provided with a mechanism for informing the client when those criteria have been violated. This allows the client to adjust its behavior for the most efficient synchronization schedule.

Second, the metadata is enhanced in four ways: Flags are added to allow the client to detect which fields are affected by which timestamps. Cross-references are added in the Table, Object, and Foreign Key metadata to facilitate retrieval of child records whose data were previously stored from the flattened representation of the main record.

Finally, the proposal adds a new transaction, called `GetSettings`. This transaction returns a dictionary of server-specific settings in somewhat the same manner as the `GetMetadata` transaction. Its initial intended use is to allow a client to query the server's throttling (bandwidth control) parameters, but it is envisioned that there will be many other uses in the future. These could include querying specific server capabilities rather than forcing a client to infer this from the version number and trial-and-error transactions.

3. Proposal

3.1. Search Transaction Changes

In Section 7.4 Optional Request Arguments, add the following:

```
Key = ".EMPTY." | NextKeyValue
```

If the client provides no value for `Key`, the search proceeds as per the current specification.

If it is set to `.EMPTY.` or a `NextKeyValue` and the request meets the server's requirements for `KEY` support as defined in the `TABLE-METADATA` for the Resource and Class, the server **MUST** provide a `<NEXTKEY>` tag with a `NextKeyValue` if it sends a `<MAXROWS>` tag in the response. For Resources deemed replicable, servers **SHOULD** support either `KeyLimit=NONE` in settings or have the resource's `Keyfield` as `KeySelect`.

If it is set to `NextKeyValue`, the server **MUST** respond with a result set that starts at an offset 1 record past the previous request and guarantees the request chain that started with an `.EMPTY.` value will not miss any records that still meet the query parameters. The results set **MAY** contain records that have been additionally modified since the beginning of a request chain that started with a `Key=.EMPTY.` argument. A `NextKeyValue` is opaque to the client, only valid once and only in the context of a single request chain. The request chain ends when the Server does not include a `<MAXROWS>` tag in the response. At the boundary of its limits, a server **MAY** send `<MAXROWS>` and `<NEXTKEY>` in a response and not deliver any records to the subsequent request (ReplyCode 20201), likewise ending the request chain.

If the client includes the `Key` request argument with a `NextKeyValue` rather than `.EMPTY.`, the client **MUST** send all the other request arguments and values used in the original request. If the client drops arguments or includes different values, the server's response **MAY** no longer be reliable. If a client issues a search for a given resource and class that does not include the `Key` optional request argument or for another resource or class, the `NextKeyValue` becomes invalid and can not be used again.

In Section 7.6, the Response Body format is modified to add the following:

```
<next-key-tag>
```

immediately before the `<RETS-REPLY>` tag. The `next-key-tag` is of the form

<NEXTKEY>NextKeyValue</NEXTKEY>

where *NextKeyValue* is an opaque string not exceeding 64 characters that may be used in the next transaction in this session as an offset to step through the result set of the query.

In addition, three new response codes are defined.

Reply Code	Meaning
20212	Invalid Key Request The transaction does not meet the server's requirements for the use of the <i>Key</i> option.
20213	Invalid Key The transaction uses a <i>Key</i> that is incorrect or is no longer valid.

3.2 Metadata Changes

3.2.1. METADATA-TABLE

Field Name	Content Type	Description
ModTimeStamp	Boolean	When true, indicates that changes to this field update the class's <i>ModTimeStamp</i> .
ForeignKey	Character	When nonblank, indicates that this field is normally populated via a foreign key. The value is the <i>ForeignKeyID</i> from the <i>METADATA-FOREIGN_KEYS</i> table.
ForeignField	Character	The <i>SystemName</i> from the child record accessed via the specified <i>ForeignKey</i> .
KeyQuery	Boolean	When true, indicates that this field may be included in a query that uses the <i>Key</i> optional argument.
KeySelect	Boolean	When true, indicates that this field may be included in the <i>Select</i> list of a query that uses the <i>Key</i> optional argument.

3.2.2. METADATA-CLASS

Field Name	Content Type	Description
ClassTimeStamp	Character	The <i>SystemName</i> of the field in the <i>METADATA-TABLE</i> that acts as the timestamp for this class.
DeletedFlagField	Character	The <i>SystemName</i> of the field in the <i>METADATA-TABLE</i> that indicates that the record is logically deleted. If this field is specified, then <i>DeletedFlagValue</i> must be specified as well.
DeletedFlagValue	Character	The value of the field designated by <i>DeletedFlagField</i> indicating that a record has been logically deleted. If the type of the field named in <i>DeletedFlagField</i> is numeric, then this value is converted to numeric type before comparing. If the type of the field named in <i>DeletedFlagField</i> is character, then the shorter of the two values is padded with blanks and the comparison made for equal length.

3.2.3. METADATA-OBJECT

Field Name	Content Type	Description
ObjectTimeStamp	Character	The <i>SystemName</i> of the field in the <i>METADATA-TABLE</i> that acts as the timestamp for objects of this type. This <i>SystemName</i> MUST appear in all classes which have objects of this type.
ObjectCount	Character	The <i>SystemName</i> of the field in the <i>METADATA-TABLE</i> that contains the count of objects of this type. This <i>SystemName</i> MUST appear in all classes which have objects of this type.

3.2.4. METADATA-FOREIGN_KEY

Field Name	Content Type	Description
ConditionalParentField	Character	The <i>SystemName</i> of a field in the parent's <i>METADATA-TABLE</i> that should be examined to determine whether this parent-child relationship should be used. If this is blank, the relationship is unconditional.
ConditionalParentValue	Character	The value of the field designated by <i>ConditionalParentField</i> indicating that this relation should be used. If the type of the field named in <i>ConditionalParentField</i> is numeric, then this value is converted to numeric type before comparing. If the type of the field named in <i>ConditionalParentField</i> is character, then the shorter of the two values is padded with blanks and the comparison made for equal length.

To determine the correct child relationship for a parent where there are multiple instances of the given *ParentResourceID/ParentClassID/ParentSystemName*, the client compares the value of the *ConditionalParentField* to the value given by *ConditionalParentValue*, and uses the first foreign key for which the values match according to the rules given in the table.

3.3. New ServerInformation transaction

This change proposal introduces an extensible mechanism for obtaining server settings. The settings may be system-wide, or may be associated with a specific resource and class.

3.3.1. Required Request Arguments

There are no required request arguments. A `GetSettings` transaction with no arguments returns system-wide settings.

3.3.2. Optional Request Arguments

<code>Resource</code>	The name of the resource for which settings are desired. A value of "*" indicates that settings for all resources should be returned.
<code>Class</code>	The name of the class for which settings are desired. A value of "*" indicates that settings for all resources should be returned. If the value of <code>Resource</code> is "*", then <code>Class</code> MUST be "*" or omitted (in which case, the asterisk is implied).

3.3.3. Response

The response is a well-formed XML document of the following form:

```
<ServerInformation>
  <Parameter name="parametername" [resource="resource ID" [class="ClassID"]]>value</Parameter>
  .
  .
  .
</ServerInformation>
```

The server MUST supply the information that applies to the `Class` level even if the information is global to the system. That is, the client is not required to infer information from the class hierarchy.

The well-known names for parameters are given below. A server extending this specification with additional parameter names MUST precede the parameter name with the string "X-".

Parameter	Level	Type	Description
<code>CurrentTimeStamp</code>	System	DateTime	The current system date and time, including time zone, in ISO 8601 format.
<code>LastTimeStamp</code>	ResourceClass	DateTime	The most recent modification timestamp of any record in the class, in ISO 8601 format.
<code>MinimumLimit</code>	ResourceClass	Numeric/String	The minimum <code>Limit</code> value for any search in this class. The value <code>NONE</code> may be returned if there is no minimum limit.
<code>KeyLimit</code>	ResourceClass	Numeric/String	The minimum <code>Limit</code> value for any search in this class that includes a <code>Key</code> optional parameter. The value <code>NONE</code> may be returned if there is no minimum limit.
<code>ReplicationSupport</code>	ResourceClass	Character	An indication of the level of replication support available for the given resource/class: <ul style="list-style-type: none"> N Replication is not supported for this resource/class: there are fields for which it is not possible to determine the last change date, or there is no support for the <code>Key</code> optional search argument. Y All fields are marked as to their controlling timestamp or foreign key, and the server supports the <code>Key</code> optional search argument. A blank query may be used to retrieve all records that the current user is permitted to access. . K All fields are marked as to their controlling timestamp or foreign key, and the server supports the <code>Key</code> optional search argument. A query MUST contain one or more of the fields marked in the metadata with the <code>KeyQuery</code> flag. .

3.3.4. Response codes

Reply Code	Meaning
20601	Not supported The transaction is not supported for the given resource/class.
20602	Miscellaneous error The transaction could not be completed; the <code>ReplyText</code> contains additional information.

4. Development Impact

Clients that wish to do accurate database replication will need to implement the optional `key` argument logic. Note that this is a new capability, so there are no compatibility issues.

The update has a moderate impact on servers that choose to implement the functionality. In addition, the new metadata for supporting the `key` function may be time-consuming to generate.

The addition of the `ServerInformation` transaction represents a mandatory enhancement for 1.6-compatible servers.

5. Compatibility

This change is backward-compatible with respect to clients: no current transactions are altered by this proposal.

RETS Change Proposal: Replace Validation Expressions

Author: Stuart Schuessler and Libor Viktorin

Organization: MarketLinx

Address: P.O. Box 24119 Knoxville, TN 37933-2119

Telephone Number: (865)218-3606

Email: sschuessler@marketlinx.com

Date: January 24, 2002

Version: 1.5

1. Synopsis

This proposal replaces the current validation expression description with a formalized BNF that is machine readable.

2. Rationale

Currently the Validation Expression section does not take into account a number of factors such as data type and format that would provide a consistent interpretation between RETS Servers. This proposal corrects this by defining the validation expression elements required to successfully determine the validity of a data element against the business rules of the system.

3. Proposal

3.1 Specification Changes

Section 11.4.9 should be rewritten as follows:

This section describes the ValidationExpression table that is referenced in Section 11.3.4. There MUST be a corresponding table entry for each ValidationExpressionID referenced in the METADATA-UPDATE_TYPES for a Resource.

The table contains expressions that are to be evaluated when a field value is entered by the user. Expressions in the list MUST be evaluated in the order in which they appear in the list. There are three types of validation expressions, each introduced by a reserved token preceding the expression:

Table 3-1 Validation Expression Data Types

Every validation expression has one of following types:

CHAR	- any string of ASCII characters
INT	- any integer (tiny, small, int, long)
FLOAT	- decimal number with fraction part
TIME	- time, date, datetime
BOOLEAN	- true or false

Table 3-2 Validation Expression Types

Keyword Type	Purpose
--------------	---------

ACCEPT Boolean If the expression is true, the field value is considered accepted without further testing. Subsequent SET expressions MUST be executed.

REJECT Boolean If the expression is true, the field value is considered rejected without further testing. Subsequent SET expressions MUST NOT be evaluated.

SET Assignment The expression MUST begin with a field name and an equal sign (“=”). The following expression is evaluated and the result stored in the designated field.

VALIDATE Assignment If the expression is true, an error MAY be returned to check if the field is valid with the client.

11.4.9.1 Validation Expression BNF

```

<Exp> ::= <NotExp>
<NotExp> ::= .NOT. <NotExp> | <OrExp>
<OrExp> ::= <AndExp> *( .OR. <AndExp> )
<AndExp> ::= <EqExp> *( .AND. <EqExp> )
<EqExp> ::= <CmpExp>
| <CmpExp> = <CmpExp>
| <CmpExp> != <CmpExp>
<CmpExp> ::= <CntExp>
| <CntExp> <= <CntExp>
| <CntExp> >= <CntExp>
| <CntExp> < <CntExp>
| <CntExp> > <CntExp>
<CntExp> ::= <SumExp> | <SumExp> .CONTAINS. <SumExp>
<SumExp> ::= <ProdExp> *( ( + | - ) <ProdExp> )
<ProdExp> ::= <AtomExp> *( ( * | / | .MOD. ) <AtomExp> )
<AtomExp> ::= ( <Exp> )
| <Value>
| <FuncExp>
<FuncExp> ::= <Func> ( <Param> *( , <Param> ) )
<Func ::= ALPHA *( ALPHANUM )
<Param ::= <Exp>
<Value ::= <SpecValue>
| <CharValue>
| <IntValue>
| <FloatValue>
| <Name>
<Name ::= ALPHA *( ALPHANUM )
<SpecValue ::= . ALPHA *( ALPHANUM ) .
<CharValue ::= ` PLAINTEXT ` | " PLAINTEXT "
<TimeValue ::= # DATE #
<IntValue ::= 0*1(+ | -) 1*(DIGITS)
<FloatValue ::= <IntValue> . *(DIGIT)

```

The text in CharValue must not include the (single or double) quote used to delimit the value.

A <Name> is a name of a field, and has a type of that field specified by the metadata.

A <TimeValue> has TIME type.

A <CharValue> has CHAR type.

A <IntValue> has INT type.

A <FloatValue> has FLOAT type.

A <SpecValue> may be one of these values:

.TRUE.	BOOLEAN
.FALSE.	BOOLEAN
.EMPTY.	CHAR
.TODAY.	TIME
.NOW.	TIME
.ENTRY.	type of the current field
.OLDVALUE.	type of the current field
.USERID.	CHAR
.USERCLASS.	CHAR
.USERLEVEL.	CHAR

.AGENTCODE.	CHAR
.BROKERCODE.	CHAR
.BROKERBRANCH.	CHAR

A <FuncExp> is a function with parameters. Following functions are defined:

Function	parameter types	type
BOOL	BOOLEAN or CHAR	BOOLEAN
CHAR	any	CHAR
TIME	TIME or CHAR	TIME
DATE	TIME or CHAR	TIME
INT	INT or FLOAT or BOOL or CHAR	INT
FLOAT	INT or FLOAT or BOOL or CHAR	FLOAT
SUBSTR	CHAR,INT,INT	CHAR
STRLEN	CHAR	INT
IIF	BOOLEAN,any,any	any

The BOOL, CHAR, TIME, DATE, INT and FLOAT functions are used just to change a type of expression. The DATE and TIME functions are synonyms.

In conversion from BOOLEAN to INT or FLOAT, .TRUE. is converted to 1 and .FALSE. is converted to 0. Casting FLOAT to INTEGER discards the fractional part. When converting to and from CHAR, functions should support the usual formats.

The SUBSTR function returns a substring of its first parameter. Second parameter is a starting position of the substring, third parameter is the ending position of the substring. Positions are 1-based.

The STRLEN function returns the length of its parameter.

The IIF function returns the value of its second parameter if the first parameter evaluates to true, or the value of its third parameter otherwise. Types of second and third parameter must be same, and it is the type of the result.

The operators may be applied on certain types, and the resulting type is defined by the following table:

Operator	Left operand	Right operand	Result
/,*,.MOD.	INT	INT	INT
/,*	INT	FLOAT	FLOAT
/,*	FLOAT	INT	FLOAT
/,*	FLOAT	FLOAT	FLOAT
+	INT	INT	INT
+	INT	FLOAT	FLOAT
+	INT or FLOAT	TIME	TIME
+	FLOAT	INT	FLOAT
+	FLOAT	FLOAT	FLOAT
+	TIME	INT or FLOAT	TIME
+	CHAR	CHAR	CHAR
-	INT	INT	INT
-	INT	FLOAT	FLOAT
-	FLOAT	INT	FLOAT
-	FLOAT	FLOAT	FLOAT

-	TIME	INT or FLOAT	TIME
-	TIME	TIME	INT
.CONTAINS.	CHAR	CHAR	CHAR
<, >, <=, >=, =, !=	Any	Same as left	BOOLEAN
.AND., .OR.	BOOLEAN	BOOLEAN	BOOLEAN
.NOT.		BOOLEAN	BOOLEAN

Additions and subtractions with times use number of seconds to represent time intervals. The INT function should support conversion from the string representing time interval into a number of seconds.

The using of strong types will help in optimizing and preprocessing the validation expressions, as well as it specifies more precisely how to calculate the expression values. However, it forced the use of conversion functions, so I added some basic other functions as well. More functions may be added to the specifications if the need shows.

A need to use numbers to represent time intervals arose from the ambiguity in expressions like

"2002-10-30" - "2002-9-10" = "P1M20D"
 "2002-10-30" - "2002-9-10" = "P50D"
 "2002-10-30" + "P1M20D" = "2002-12-20"
 "2002-10-30" + "P50D" = "2002-12-19"

The Validation Expression metadata starts with a <METADATA-VALIDATION_EXPRESSION> tag with Resource, Version and Date attributes. This is followed by a <COLUMNS> section, which contains the name of the fields as defined fields as defined in Table 11-19, followed by the <DATA> section, which contains the actual Validation Expressions. The Validation Expression metadata has the following format:

```

<METADATA-VALIDATION_EXPRESSION · Resource=" resource-id "
· Version=" validation-expression-type-version " ·
Date=" validation-expression-type-date "> ↓ <COLUMNS>→ validation-
expression-type-field
*( → validation-expression-type-field )→</COLUMNS> ↓
*( <DATA>→ validation-expression-type-data *(→ validation-expression-type-data )→
</DATA> ↓)
</METADATA-VALIDATION_EXPRESSION>
↓ resource-id ::= 1*32ALPHANUM

```

This value MUST be a ResourceID found in the Resource metadata. It is the Resource to which the Objects belong. *validation-expression-version*::= 1*2DIGITS . 1*2DIGITS . 1*5DIGITS

This is the version number of the Validation Expression metadata. The convention used is a “<major>.<minor>.<release>” numbering scheme. Every time the Validation

Expression metadata changes the version number should be increased. *valid-expression-date:=DATE*

The latest change date of the ValidationExpression metadata.

valid-expression-field:=<Field Name from Table 11-20>

valid-expression-data:=<expression type keyword><valid validation expression>

Table 11-21 Metadata Content: Validation Expression

Field Name	Content Type	Description
Validation-ExpressionID	1*32ALPHANUM	A unique ID for the ValidationExpression. This ID is referenced as the ValidationExpression in Section 11.3.4.
Validation-ExpressionType Value	1*32ALPHANUM	A validation expression type from Table 11-17.
	1*512TEXT	The test expression to be evaluated.

An example Validation Expression section follows:

GetMetadata request:

```
Type:
METADATA-
VALIDATION_
EXPRESSION
ID:
Property
```

Compact reply:

```
<METADATA-VALIDATION_EXPRESSION Resource="PROPERTY"
Version="1.00.00000" Date="Tuesday, 03-Oct-2000 17:10:53 GMT">
<COLUMNS> ValidationExpressionID ValidationExpressionType
Value </COLUMNS>
<DATA> 10010 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10011 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10012 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10013 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10014 REJECT .ENTRY. < (ListDate + #P90D#) </DATA>
<DATA> 10015 REJECT ((ListStatus = 'EXP') .OR. (ListStatus = 'CLOSD')
.OR. (ListStatus = 'WITH')) .AND. (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10016 REJECT IIF ((ListStatus='ACT') .AND. (.USERCLASS. !=
'STAFF'), (.ENTRY. < .OLDVALUE.), .FALSE.) </DATA>
<DATA> 10017 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. > .TODAY.,
.FALSE.) </DATA>
<DATA> 10018 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. < (.TODAY. -
#P14D#), .FALSE.) </DATA>
<DATA> 10019 REJECT (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10020 SET ListPriceOrig = .ENTRY. </DATA>
<DATA> 10021 SET ListPriceOrig = IIF (.OLDVALUE. =
.EMPTY., .ENTRY., ListPriceOrig) </DATA>
<DATA> 10022 REJECT .USERCLASS. != 'STAFF' </DATA>
<DATA> 10023 SET ListStatus = 'ACT' </DATA>
<DATA> 10024 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'WITH')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10025 REJECT IIF((.ENTRY. = 'WITH'), (.USERCLASS. != 'STAFF'),
.FALSE.)) </DATA>
<DATA> 10026 SET WithdrawnDate = IIF(((.ENTRY. = 'WITH') .AND.
(WithdrawnDate = .EMPTY.)), .TODAY., WithdrawnDate) </DATA>
<DATA> 10027 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'EXP')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
```

```

<DATA> 10028 REJECT IIF((.ENTRY. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10029 REJECT IIF(((.ENTRY. = 'CONKO') .OR. (.ENTRY.= 'CONTG')
.OR. (.ENTRY.= 'PEND').OR.(.ENTRY.= 'LPRCH')), ((Terms=.EMPTY.) .OR.
(PendingDate=.EMPTY.).OR.(OfficeSell = .EMPTY.).OR.(AgentSell =
.EMPTY.).OR.(ClosedDateProposed = .EMPTY.)), (.FALSE.)) </DATA>
<DATA> 10030 REJECT IIF((.OLDVALUE. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10031 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10032 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10033 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10034 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10035 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10036 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10037 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10038 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10039 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10040 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10041 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10042 REJECT .ENTRY. < (ListDate + #P90D#) </DATA>
<DATA> 10043 REJECT ((ListStatus = 'EXP') .OR. (ListStatus = 'CLOSD')
.OR. (ListStatus = 'WITH')) .AND. (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10044 REJECT IIF ((ListStatus='ACT') .AND. (.USERCLASS. !=
'STAFF'), (.ENTRY. < .OLDVALUE.), .FALSE.) </DATA>
<DATA> 10045 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. < (.TODAY. -
#P14D#), .FALSE.) </DATA>
<DATA> 10046 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. > .TODAY.,
.FALSE.) </DATA>
<DATA> 10047 REJECT (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10048 SET ListPriceOrig = .ENTRY. </DATA>
<DATA> 10049 SET ListPriceOrig = IIF (.OLDVALUE. =
.EMPTY., .ENTRY., ListPriceOrig) </DATA>
<DATA> 10050 REJECT .USERCLASS. != 'STAFF' </DATA>
<DATA> 10051 SET ListStatus = 'ACT' </DATA>
<DATA> 10052 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'WITH')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10053 REJECT IIF((.ENTRY. = 'WITH'), (.USERCLASS. != 'STAFF'),
(.FALSE.)) </DATA>
<DATA> 10054 SET WithdrawnDate = IIF(((.ENTRY. = 'WITH') .AND.
(WithdrawnDate = .EMPTY.)), .TODAY., WidthdrawnDate) </DATA>
<DATA> 10055 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'EXP')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10056 REJECT IIF((.ENTRY. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10057 REJECT IIF(((.ENTRY. = 'CONKO') .OR. (.ENTRY.= 'CONTG')
.OR. (.ENTRY.= 'PEND').OR.(.ENTRY.= 'LPRCH')), ((Terms=.EMPTY.) .OR.
(PendingDate=.EMPTY.).OR.(OfficeSell = .EMPTY.).OR.(AgentSell =
.EMPTY.).OR.(ClosedDateProposed = .EMPTY.)), (.FALSE.)) </DATA>
<DATA> 10058 REJECT IIF((.OLDVALUE. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10059 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10060 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10061 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10062 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>

```

```

<DATA> 10063 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10064 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10065 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10066 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10067 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10068 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10069 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10070 REJECT .ENTRY. < (ListDate + #P90D#) </DATA>
<DATA> 10071 REJECT ((ListStatus = 'EXP') .OR. (ListStatus = 'CLOSD')
.OR. (ListStatus = 'WITH')) .AND. (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10072 REJECT IIF ((ListStatus='ACT') .AND. (.USERCLASS. !=
'STAFF'), (.ENTRY. < .OLDVALUE.), .FALSE.) </DATA>
<DATA> 10073 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. < (.TODAY. -
#P14D#), .FALSE.) </DATA>
<DATA> 10074 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. > .TODAY.,
.FALSE.) </DATA>
<DATA> 10075 REJECT (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10076 SET ListPriceOrig = .ENTRY. </DATA>
<DATA> 10077 SET ListPriceOrig = IIF (.OLDVALUE. =
.EMPTY., .ENTRY., ListPriceOrig) </DATA>
<DATA> 10078 REJECT .USERCLASS. != 'STAFF' </DATA>
<DATA> 10079 SET ListStatus = 'ACT' </DATA>
<DATA> 10080 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'WITH')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10081 REJECT IIF((.ENTRY. = 'WITH'), (.USERCLASS. != 'STAFF'),
(.FALSE.)) </DATA>
<DATA> 10082 SET WithdrawnDate = IIF(((.ENTRY. = 'WITH') .AND.
(WithdrawnDate = .EMPTY.)), .TODAY., WithdrawnDate) </DATA>
<DATA> 10083 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'EXP')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10084 REJECT IIF((.ENTRY. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10085 REJECT IIF(((.ENTRY. = 'CONKO') .OR. (.ENTRY.= 'CONTG')
.OR. (.ENTRY.= 'PEND').OR. (.ENTRY.= 'LPRCH')), ((Terms=.EMPTY.) .OR.
(PendingDate=.EMPTY.).OR.(OfficeSell = .EMPTY.).OR.(AgentSell =
.EMPTY.).OR.(ClosedDateProposed = .EMPTY.)), (.FALSE.)) </DATA>
<DATA> 10086 REJECT IIF((.OLDVALUE. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10087 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10088 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10089 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10090 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10091 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10092 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10093 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10094 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10095 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10096 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10097 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10098 REJECT .ENTRY. < (ListDate + #P90D#) </DATA>

```

```

<DATA> 10099 REJECT ((ListStatus = 'EXP') .OR. (ListStatus = 'CLOSD')
.OR. (ListStatus = 'WITH')) .AND. (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10100 REJECT IIF ((ListStatus='ACT') .AND. (.USERCLASS. !=
'STAFF'), (.ENTRY. < .OLDVALUE.), .FALSE.) </DATA>
<DATA> 10101 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. < (.TODAY. -
#P14D#), .FALSE.) </DATA>
<DATA> 10102 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. > .TODAY.,
.FALSE.) </DATA>
<DATA> 10103 REJECT (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10104 SET ListPriceOrig = .ENTRY. </DATA>
<DATA> 10105 SET ListPriceOrig = IIF (.OLDVALUE. =
.EMPTY., .ENTRY., ListPriceOrig) </DATA>
<DATA> 10106 REJECT .USERCLASS. != 'STAFF' </DATA>
<DATA> 10107 SET ListStatus = 'ACT' </DATA>
<DATA> 10108 REJECT IIF((.ENTRY. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10109 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'WITH')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10110 REJECT IIF((.ENTRY. = 'WITH'), (.USERCLASS. != 'STAFF'),
(.FALSE.)) </DATA>
<DATA> 10111 SET WithdrawnDate = IIF(((.ENTRY. = 'WITH') .AND.
(WithdrawnDate = .EMPTY.)), .TODAY., WidthdrawnDate) </DATA>
<DATA> 10112 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'EXP')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10113 REJECT IIF(((.ENTRY. = 'CONKO') .OR. (.ENTRY.= 'CONTG')
.OR. (.ENTRY.= 'PEND').OR. (.ENTRY.= 'LPRCH')), ((Terms=.EMPTY.) .OR.
(PendingDate=.EMPTY.) .OR. (OfficeSell = .EMPTY.) .OR. (AgentSell =
.EMPTY.) .OR. (ClosedDateProposed = .EMPTY.)), (.FALSE.)) </DATA>
<DATA> 10114 REJECT IIF((.OLDVALUE. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10115 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10116 REJECT (.ENTRY. > (.TODAY. + #PLY#)) </DATA>
<DATA> 10117 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10118 REJECT (.ENTRY. > (.TODAY. + #PLY#)) </DATA>
<DATA> 10119 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10120 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10121 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10122 REJECT (.ENTRY. > (.TODAY. + #PLY#)) </DATA>
<DATA> 10123 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10124 REJECT (.ENTRY. > (.TODAY. + #PLY#)) </DATA>
<DATA> 10125 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10126 REJECT .ENTRY. < (ListDate + #P90D#) </DATA>
<DATA> 10127 REJECT ((ListStatus = 'EXP') .OR. (ListStatus = 'CLOSD')
.OR. (ListStatus = 'WITH')) .AND. (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10128 REJECT IIF ((ListStatus='ACT') .AND. (.USERCLASS. !=
'STAFF'), (.ENTRY. < .OLDVALUE.), .FALSE.) </DATA>
<DATA> 10129 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. < (.TODAY. -
#P14D#), .FALSE.) </DATA>
<DATA> 10130 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. > .TODAY.,
.FALSE.) </DATA>
<DATA> 10131 REJECT (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10132 SET ListPriceOrig = .ENTRY. </DATA>
<DATA> 10133 SET ListPriceOrig = IIF (.OLDVALUE. =
.EMPTY., .ENTRY., ListPriceOrig) </DATA>

```

```

<DATA> 10134 REJECT .USERCLASS. != 'STAFF' </DATA>
<DATA> 10135 SET ListStatus = 'ACT' </DATA>
<DATA> 10136 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'WITH')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10137 REJECT IIF((.ENTRY. = 'WITH'), (.USERCLASS. != 'STAFF'),
(.FALSE.)) </DATA>
<DATA> 10138 SET WithdrawnDate = IIF(((.ENTRY. = 'WITH') .AND.
(WithdrawnDate = .EMPTY.)), .TODAY., WidthdrawnDate) </DATA>
<DATA> 10139 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'EXP')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10140 REJECT IIF((.ENTRY. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10141 REJECT IIF(((.ENTRY. = 'CONKO') .OR. (.ENTRY.= 'CONTG')
.OR. (.ENTRY.= 'PEND').OR.(.ENTRY.= 'LPRCH')), ((Terms=.EMPTY.) .OR.
(PendingDate=.EMPTY.).OR.(OfficeSell = .EMPTY.).OR.(AgentSell =
.EMPTY.).OR.(ClosedDateProposed = .EMPTY.)), (.FALSE.)) </DATA>
<DATA> 10142 REJECT IIF((.OLDVALUE. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10143 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10144 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10145 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10146 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10147 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10148 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10149 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10150 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10151 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10152 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10153 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10154 REJECT .ENTRY. < (ListDate + #P90D#) </DATA>
<DATA> 10155 REJECT ((ListStatus = 'EXP') .OR. (ListStatus = 'CLOSD')
.OR. (ListStatus = 'WITH')) .AND. (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10156 REJECT IIF ((ListStatus='ACT') .AND. (.USERCLASS. !=
'STAFF'), (.ENTRY. < .OLDVALUE.), .FALSE.) </DATA>
<DATA> 10157 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. > .TODAY.,
.FALSE.) </DATA>
<DATA> 10158 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. < (.TODAY. -
#P14D#), .FALSE.) </DATA>
<DATA> 10159 REJECT (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10160 SET ListPriceOrig = .ENTRY. </DATA>
<DATA> 10161 SET ListPriceOrig = IIF (.OLDVALUE. =
.EMPTY., .ENTRY., ListPriceOrig) </DATA>
<DATA> 10162 REJECT .USERCLASS. != 'STAFF' </DATA>
<DATA> 10163 SET ListStatus = 'ACT' </DATA>
<DATA> 10164 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'WITH')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10165 REJECT IIF((.ENTRY. = 'WITH'), (.USERCLASS. != 'STAFF'),
(.FALSE.)) </DATA>
<DATA> 10166 SET WithdrawnDate = IIF(((.ENTRY. = 'WITH') .AND.
(WithdrawnDate = .EMPTY.)), .TODAY., WidthdrawnDate) </DATA>
<DATA> 10167 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'EXP')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10168 REJECT IIF((.ENTRY. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>

```

```

<DATA> 10169 REJECT IIF(((.ENTRY. = 'CONKO') .OR. (.ENTRY.='CONTG')
.OR. (.ENTRY.='PEND').OR.(.ENTRY.='LPRCH')), ((Terms=.EMPTY.) .OR.
(PendingDate=.EMPTY.).OR.(OfficeSell = .EMPTY.).OR.(AgentSell =
.EMPTY.).OR.(ClosedDateProposed = .EMPTY.)), (.FALSE.)) </DATA>
<DATA> 10170 REJECT IIF((.OLDVALUE. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10171 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10172 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10173 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10174 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10175 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10176 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10177 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10178 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10179 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10180 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10181 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10182 REJECT .ENTRY. < (ListDate + #P90D#) </DATA>
<DATA> 10183 REJECT ((ListStatus = 'EXP') .OR. (ListStatus = 'CLOSD')
.OR. (ListStatus = 'WITH')) .AND. (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10184 REJECT IIF ((ListStatus='ACT') .AND. (.USERCLASS. !=
'STAFF'), (.ENTRY. < .OLDVALUE.), .FALSE.) </DATA>
<DATA> 10185 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. < (.TODAY. -
#P14D#), .FALSE.) </DATA>
<DATA> 10186 REJECT IIF (.USERCLASS. != 'STAFF', .ENTRY. > .TODAY.,
.FALSE.) </DATA>
<DATA> 10187 REJECT (.USERCLASS. != 'STAFF') </DATA>
<DATA> 10188 REJECT .USERCLASS. != 'STAFF' </DATA>
<DATA> 10189 SET ListStatus = 'ACT' </DATA>
<DATA> 10190 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'WITH')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10191 REJECT IIF((.ENTRY. = 'WITH'), (.USERCLASS. != 'STAFF'),
(.FALSE.)) </DATA>
<DATA> 10192 SET WithdrawnDate = IIF(((.ENTRY. = 'WITH') .AND.
(WithdrawnDate = .EMPTY.)), .TODAY., WidthdrawnDate) </DATA>
<DATA> 10193 REJECT IIF(((.ENTRY. = 'ACT') .AND. (.OLDVALUE. =
'EXP')), (.USERCLASS. != 'STAFF'), (.FALSE.)) </DATA>
<DATA> 10194 REJECT IIF((.ENTRY. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10195 REJECT IIF(((.ENTRY. = 'CONKO') .OR. (.ENTRY.='CONTG')
.OR. (.ENTRY.='PEND').OR.(.ENTRY.='LPRCH')), ((Terms=.EMPTY.) .OR.
(PendingDate=.EMPTY.).OR.(OfficeSell = .EMPTY.).OR.(AgentSell =
.EMPTY.).OR.(ClosedDateProposed = .EMPTY.)), (.FALSE.)) </DATA>
<DATA> 10196 REJECT IIF((.OLDVALUE. = 'CLOSD'), (.USERCLASS. !=
'STAFF'), (.FALSE.)) </DATA>
<DATA> 10197 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10198 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10199 REJECT (.ENTRY. < ListDate) </DATA>
<DATA> 10200 REJECT (.ENTRY. > (.TODAY. + #P1Y#)) </DATA>
<DATA> 10201 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
<DATA> 10202 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>

```

```
<DATA> 10203 REJECT (.USERCLASS. != 'STAFF') .AND. (.TODAY. >
(ListDate + #P30D#)) </DATA>
</METADATA-VALIDATION_EXPRESSION>
```

4. Development Impact

Because the proposal changes the validation expressions as they are currently defined it will have a big impact on current client and server software not currently following this format.

5. Compatibility

This is not backward compatible.

Last changed 11 March, 2003, 1815 EST by [Sstuart Schuessler](#).