

1.4 LSX LC

Nachdem in den letzten beiden Kapiteln zwei Möglichkeiten erläutert wurden, um Daten in einer heterogenen Systemlandschaft über konfigurierbare Dokumente auf relativ einfache Art und Weise abzugleichen und insbesondere auch RealTime-Zugriffe zu ermöglichen, geht es in diesem Kapitel um den programmierten Zugriff auf Datenquellen. Auf den ersten Blick scheint es nicht notwendig zu sein, zwischen verschiedenen Systemen eine Verbindung über eine objektbasierte beziehungsweise objektorientierte Programmiersprache herzustellen, wenn dies auch wesentlich einfacher und zum Teil auch schneller über Programme wie den LEI oder DECS zu realisieren ist. Leider stößt man aber immer wieder an die Grenzen dieser Methoden, so daß man sich eine flexiblere Möglichkeit für den Verbindungsaufbau, die Datenextraktion und die Datenbearbeitung wünscht. Diese Grenzen werden dann besonders deutlich, wenn man die Ergebnisse, die der LEI oder DECS liefern, direkt weiterverarbeiten oder gleich anschließend eine Auswertung der Informationen durchführen möchte.

An dieser Stelle können die *LotusScript Extensions for Lotus Domino Connectors (LSX LC)* sehr einfach in die bestehende Entwicklungsumgebung von Lotus Notes/Domino integriert und eingesetzt werden.

Die *LSX LC* erweitert den Funktionsumfang der LotusScript-Sprache um eine Reihe von Klassen und Methoden, mit denen objektbasierter Zugriff auf externe Datensysteme ermöglicht wird. Sie stellen einen weiteren wichtigen Teil der *Lotus Enterprise Integration* dar, da sie über die definierten Schnittstellen des modularen Schichtenmodells verfügen. Durch diese Integration wird jede Datenverbindung zu einer externen Datenquelle über einen zugehörigen Konnektor hergestellt und ein bidirektionaler Datenaustausch zwischen dem externen System und einem Dominosystem über die definierten Schnittstellen durchgeführt. Obwohl jeder installierte Konnektor über die Script-Erweiterungen individuell mit den durch ihn bereitgestellten Parametern angesprochen werden muß, sind die zugrundeliegenden Datenobjekte in ihrer Art statisch und von dem jeweiligen Verbindungstool unabhängig. Dem Entwickler wird also nicht für jeden Konnektor eine eigene Schnittstelle mit individuellen Parametern und Objekten zugemutet, er kann vielmehr für jeden Zugriff auf die gleiche Basis zurückgreifen. Ein optimal entwickeltes Programm kann über Parameter in der Art gesteuert werden, daß es für eine Vielzahl von Konnektoren und damit für mehrere Backend-Datensysteme einsetzbar ist. Das *LSX*



LC-Objektmodell wird durch den Domino-Server ab Release 4.63 und durch den Lotus Enterprise Integrator unterstützt.

In allen Applikationen, die auf Basis von Lotus Notes beziehungsweise Lotus Domino entwickelt werden, kann mit Hilfe der *LSX LC* eine individuelle Schnittstelle zu externen Datenquellen integriert werden. Da der Zugriff auf diese Applikationen programmtechnisch sowohl für einen Notes-Client als auch für einen Web-Browser ermöglicht werden kann, entsteht durch die Entwicklung individueller und standardisierter Software mit Hilfe des *LSX LC* eine einfache und einheitliche Plattform, um unternehmensweit Daten zu verbreiten und zu bearbeiten.

Das Objektmodell der *LSX LC* kann sowohl als eigenständige Erweiterung in die bestehende LotusScript-Umgebung integriert oder auch mit den Möglichkeiten des LEI kombiniert werden.

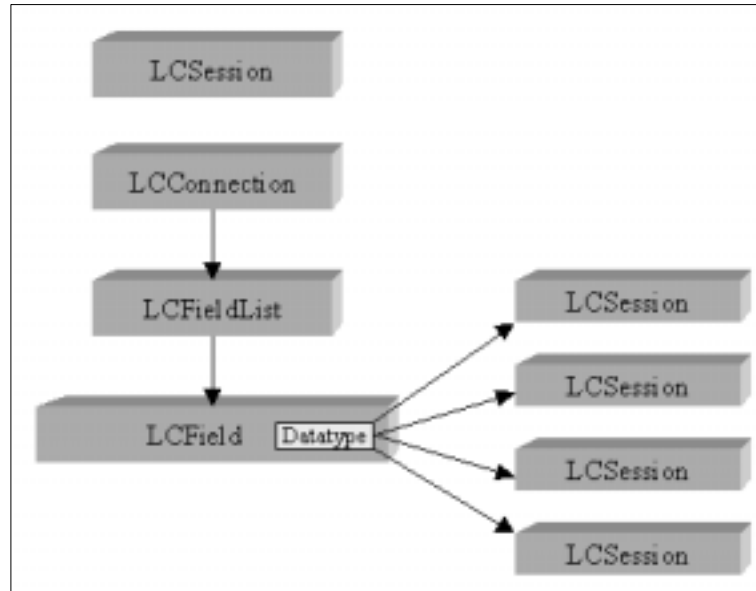
1.4.1 Das Objektmodell LSX LC

Die *LSX LC* erweitert den Funktionsumfang des bestehenden Objektmodells von LotusScript um die vier Klassen *LCSession*, *LCConnection*, *LCFieldlist* und *LCField*. Weiterhin stehen die vier zusätzliche Datentypen *LCStream*, *LCDatetime*, *LCNumeric* und *LCCurrency* zur Verfügung. Die folgende Abbildung zeigt die Beziehungen zwischen den verschiedenen Klassen und Datentypen.

Die neuen Objekte werden in der Referenzliste der möglichen Funktionen der LotusScript Entwicklungsumgebung aufgelistet, wenn die *LSX LC* mit dem Aufruf *Useslx "*lsxlc"* in den aktuellen Programmcode eingebunden wurde.

In den folgenden Beispielcodes wird an verschiedenen Stellen eine Fehlerbehandlung vorgestellt und beschrieben. Diesen Routinen und Prüfungen sollte in jedem Fall eine wichtige Rolle in der Entwicklung von Programmen mit dem *LSX LC* zuteil werden. Routinierten Entwicklern erscheint dieser Hinweis sicherlich überflüssig, denn wer einmal nach einem undefinierten Fehler in einem Programmscript gesucht hat, weiß, wie aufwendig eine derartige Suche sein kann. Im späteren Verlauf dieses Buches wird in Kapitel über das *LSX R/3* dieser Ratsschlag noch einmal deutlich angesprochen und anhand von Beispielcodes beschrieben.

Abb. 1.57:
Beziehungsmodell zwischen
LSX-LC-Klassen



LCSession

Die Klasse *LCSession* ist für einen Zugriff auf externe Datenquellen nicht notwendig. Sie muß weder definiert sein, noch leiten sich aus ihr irgendwelche weiteren Objekte ab.

Es ist aber empfehlenswert eine *LCSession* einzurichten, da sie während des Programmablaufs wichtige Informationen zu eventuell auftretenden Laufzeitfehlern liefert. Diese lassen sich über geeignete Errorhandler abfragen, abfangen, ausgeben oder umgehen. Weiterhin stellt die Klasse *LCSession* die Möglichkeit bereit, eine Liste der aktuell zur Verfügung stehenden Konnektoren zu erstellen.

Abb. 1.58: Beispielcode für
LCSession

```

Option Public
Option Declare
UseSx "*1sx1c"

Sub Initialize

    '+++ Neue Instanz vom Objekt LCSession
    Dim thislcsession As New LCSession
    '+++ Parameter der Konnektoren
    Dim parvalue As New LCStream
    Dim parname As New LCStream
    Dim concode As Long
    Dim conname As String
    
```



```
'+++ Fehlerbehandlung EIN
On Error Goto MyErrorHandler

'+++ Überprüfen ob der ODBC-Konnektor installiert ist.
conname = "odbc2"
If thislcsession.LookupConnector(conname, concode, parvalue, parname) Then
    '+++ OK, Connector installiert -> Ausgabe der Konnektor-Parameter
    Print "ODBC-Connector mit Code " & Cstr(concode) & ". Parameter " & _
        parname.text & " mit folgenden Werten " & parvalue.text
Else
    MsgBox "ODBC-Connector nicht installiert !"
End
End If

'+++ Verarbeitung für 5 Sekunden anhalten
thislcsession.sleep(5000)

'+++ Ausgabe aller installierten Konnektoren
'+++ LCLIST_FIRST liefert das erste Element der Liste zurück
'+++ LCLIST_NEXT liefert das nächste Element der Liste.
'+++ Bei erster Verwendung liefert LCLIST_NEXT das erste Element.
If thislcsession.ListConnector(LCLIST_FIRST, conname, concode, parvalue, parname) Then
    Print Ucase(conname) & "-Connector mit Code " & Cstr(concode) & ". Parameter " & _
        parname.text & " mit folgenden Werten " & parvalue.text
    While session.ListConnector(LCLIST_NEXT, conname, concode, parvalue, parname)
        Print Ucase(conname) & "-Connector mit Code " & Cstr(concode) & ". Parameter " & _
            parname.text & " mit folgenden Werten " & parvalue.text
    Wend
End If

End
MyErrorHandler:

Dim status As Integer
Dim result As String
Dim extmsgcode As Long
Dim extmsg As String

'+++ Liegt der Fehler im LCLSX ?
If thislcsession.status <> LCSUCCESS Then
    '+++ Aktuellen Status der LCSession lesen
    status = thislcsession.GetStatus (result, extmsgcode, extmsg)
    '+++ Liegt der Fehler in der externen Datenquelle?
    If (thislcsession.status = LCFAIL_EXTERNAL)Then
        '+++ Fehler liegt in der externen Datenquelle ->
        'Externen Fehlercode und Meldung ausgeben
        Print "ODBC-Fehler " & Cstr(extmsgcode) & ": " & extmsg
    Else
        '+++ Fehler wurde durch Konnektor ausgelöst ->
        'Fehlermeldung des Konnektors ausgeben
        Print "ODBC-Connector Fehler: " & result
    End If
    '+++ Status der LCSession zurücksetzen
    thislcsession.clearstatus
Else
    '+++ Kein LCLSX-Fehler -> Allgemeine Fehlermeldung ausgeben
    Print "Allgemeiner Fehler " & Err() & ": " & Error()
End If

End Sub
```

LCConnection

Mit Hilfe der Klasse *LCConnection* erzeugt man eine neue Instanz für einen *Lotus Domino Connector*. Durch dieses Objekt ist es dann möglich, auf die externe Datenquelle zuzugreifen.

Die Eigenschaften der neuen Instanz hängen von dem jeweiligen Connector ab, für den diese Connection abgeleitet wurde. Für einen eingebundenen Connector können mehrere Objekte vom Typ *LCConnection* erzeugt werden.

Abb. 1.59:
Beispielcode für
LCConnection

```
Option Public
Option Declare
UseIsx "*/sx1c"

Sub Initialize

    '+++ Neue Instanz vom Objekt LCSession
    Dim thisLCSession As New LCSession
    '+++ Neues Verbindungsobjekt vom Typ ODBC erstellen
    Dim ODBCConnection As New LCConnection ("odbc2")
    '+++ Objekt vom Typ LCFieldList nimmt die Ergebnisliste der Anfrage an
    '+++ die Datenbank auf
    Dim ResultList As New LCFieldList
    Dim ResultField As LCField
    Dim Fieldposition As Long

    '+++ Fehlerbehandlung EIN
    On Error Goto MyErrorHandler

    '+++ Verbindungsparameter zur ODBC-Schnittstelle festlegen
    '+++ Die Angaben unterscheiden sich für die verschiedenen Konnektoren.
    ODBCConnection.Database = "DatasourceMitarbeiter-Stammdaten" ' ODBC-Datenquelle
    ODBCConnection.UserID = "User_Personal001" '++ Gültige UserID
    ODBCConnection.Password = "passwort" '++ Gültiges Passwort
    '+++ Verbindung herstellen
    ODBCConnection.Connect

    '+++ Selektionsstring aufbauen.
    selectstring = "SELECT * FROM Mitarbeiter-Stammdaten"

    '+++ Auf der angegebenen ODBC-DB wird die Selektion ausgeführt.
    '+++ Das Ergebnis wird in den Objekten vom Typ LCField gespeichert,
    '+++ die über die ResultList verwaltet werden.
    If ODBCConnection.Execute(selectstring, ResultList) = 0 Then
        Print "Es wurden keine Dokumente gefunden, die der Suchbedingung " & _
            " entsprechen"
    End If
    End If

    '+++ Alle Werte der Spalte Vorname werden aus der Ergebnisliste entfernt.
    Call ResultList.LookUp("Vorname", Fieldposition)
    Call ResultList.Remove(Fieldposition)

    '+++ In diesem Fall sollen nur die Nachnamen der Ergebnisrecords
    '+++ ausgegeben werden.
    Set ResultField = ResultList.LookUp("Nachname")
    If Fieldposition > 0 Then
        Print "Feld Nachname in dem Ergebnissatz an Position " & _
```



```

        Cstr(Fieldposition) & " gefunden!"
    '+++ Es werden alle gefundenen Datensätze aus der Ergebnisliste ausgelesen
    While ODBCConnection.Fetch(ResultList) > 0
        Print ResultField.Text(0)
    Wend
Else
    Print "Feld Nachname in dem Ergebnissatz nicht gefunden ..."
End
End If

'+++ Verbindung zur ODBC-Datenquelle wird geschlossen
ODBCConnection.Disconnect

End

MyErrorHandler:

'+++ Hier wird die Fehlerbehandlung ausgeführt.
'+++ Siehe dazu LCSession

End Sub

```

LCFieldlist

Objekte dieser Klasse repräsentieren Metainformationen, die aus Datensätzen (Records) einer externen Datenquelle bestehen. Sie sind das Resultat einer Anfrage an die angegebene Datenquelle. Eine *LCFieldlist* ist eine Liste von Objektbeschreibungen und Verweisen auf Objekte vom Typ *LCField*. Diese wiederum können aus einem oder mehreren Ergebnisrecords bestehen.

Die Elemente einer *LCFieldlist* können automatisch generiert werden, indem die abgeleitete Variable vor der Datenaufnahme ohne eine Vorgabestruktur initialisiert wird. Ist dies nicht gewünscht, kann durch die Vorbelegung der deklarierten *LCFieldlist* mit festen Datentypen eine Datentransformation oder auch Datenselektion stattfinden.

Die Werte einer *LCFieldlist*, also Objekte vom Typ *LCField*, können gelesen und modifiziert werden, neue Elemente können in eine Liste hinzugefügt und Elemente aus einer Liste gelöscht werden.

Mit den Methoden *Map*, *MapName*, *Merge* und *MergeVirtual* können aus einer Ergebnisliste virtuelle Listen erzeugt werden, die auf die Elemente der Ursprungsliste verweisen. Dadurch ist es möglich, daß Listenelemente in der abgeleiteten Liste umsortiert und umbenannt werden können, ohne die ursprünglichen Instanzen der Ergebnisliste zu verändern. Weiterhin können die Objekte einer Liste mit den Objekten einer anderen gemischt und in einer dritten abgelegt werden. Mit diesem Ergebnis kann dann im folgenden zum Beispiel eine sortierte Gesamtausgabe von Resultaten aus verschiedenen Datenquellen erstellt werden.



Abb. 1.60: Option Public
Beispielcode für
LCFieldlist

```
Option Public
Option Declare
Uselsx "xlxl"

Sub Initialize

    '+++ Neue Instanz vom Objekt LCSession
    Dim thislcsession As New LCSession
    '+++ Neues Verbindungsobjekt vom Typ ODBC erstellen
    Dim ODBCConnection As New LCConnection ("odbc2")
    '+++ Objekt vom Typ LCFieldlist nimmt die Ergebnisliste der Anfrage
    '+++ an die Datenbank auf
    Dim thisfieldlist As New LCFieldList
    Dim counter As Long
    Dim thisresultfield As LCField
    Dim fieldpos As Long

    '+++ Fehlerbehandlung EIN
    On Error Goto MyErrorHandler

    '+++ Verbindungsparameter zur ODBC-Schnittstelle festlegen und
    '+++ Verbindung aufbauen
    ODBCConnection.Database = "DatasourceMitarbeiter-Stammdaten"
    ODBCConnection.Userid = "User_Personal001"
    ODBCConnection.Password = "passwort"
    ODBCConnection.Connect

    '+++ Selektionsstring aufbauen und Datensätze aus der externen Datenquelle
    '+++ in die Variable thisfieldlist einlesen
    selectstring = "SELECT * FROM Mitarbeiter-Stammdaten"
    If ODBCConnection.Execute(selectstring, thisfieldlist) = 0 Then
        '+++ Keine Ergebniswerte ...
        Print "Es wurden keine Dokumente gefunden, die der Suchbedingung " + _
            "entsprechen"
    End
    End If

    '+++ Position eines Feldes in der Ergebnisliste feststellen.
    Call thisfieldlist.LookUp("Vorname", fieldpos)
    Call thisfieldlist.Remove(Fieldposition)

    '+++ Alle Daten der Ergebnisliste ausgeben
    '+++ Erste Datenzeile ergibt die Spaltennamen
    counter = ODBCConnection.Fetch(thisfieldlist, 1, 1)
    While counter > 0
        Print ResultField.Text(0)
        counter = ODBCConnection.Fetch(thisfieldlist, 1, 1)
    Wend

    ODBCConnection.Disconnect

End

MyErrorHandler:

    '+++ Hier wird die Fehlerbehandlung ausgeführt.
    '+++ Siehe dazu LCSession

End Sub
```



LCField

LCField ist ein Container für Daten und Informationen. In einem Objekt dieses Typs können entweder einer oder mehrere Werte des gleichen definierten Datentyps gespeichert werden.

Als Datentypen für die Klasse *LCField* können die erweiterten Datentypen der *LSX LC* verwendet werden. Weiterhin können es Elemente vom Typ *long integer*, *double precision floating point* und in besonderen Fällen auch Elemente vom Typ *LCFieldlist* oder *LCConnection* sein.

```
Option Public
Option Declare
UseIsx "*/lsxlc"
```

```
Sub Initialize
```

```
    Dim thisLcSession As New LCSession
    Dim ODBCConnection As New LCConnection ("odbc2")
    Dim ResultList As New LCFieldList
    Dim ResultField As LCField
    Dim ResultFieldCurrency As New LCField (LCTYPE_CURRENCY, 1)
    Dim Fieldposition As Long
    Dim emp_loan As New LCCurrency
    Dim emp_newloan As New LCCurrency
    Dim emp_text As New LCStream
```

```
    On Error Goto MyErrorHandler
```

```
    ODBCConnection.Database = "DatasourceMitarbeiter-Stammdaten"
    ODBCConnection.UserID = "User_Personal001"
    ODBCConnection.Password = "passwort"
    ODBCConnection.Connect
```

```
    selectstring = "SELECT * FROM Mitarbeiter-Stammdaten"
    If ODBCConnection.Execute(selectstring, ResultList) = 0 Then
        Print "Es wurden keine Dokumente gefunden, die der " + _
            "Suchbedingung entsprechen"
    End
End If
```

```
    '+++ Ergebnisliste für einen Eintrag durchlaufen und
    '+++ Mitarbeiterdaten ausgeben
    counter = ODBCConnection.Fetch (ResultList, 1, 1)
    '+++ FieldCount gibt die Anzahl der Elemente in einem Datensatz an
    For index = 1 To ResultList.FieldCount
        '+++ Alle Datensatzelemente werden nacheinander gelesen
        Set ResultField = ResultList.GetField(index)
        With ResultField
            '+++ Der Datentyp des aktuellen Feldes wird bestimmt. Hier werden nur
            '+++ die Felder vom Datentyp TEXT und CURRENCY betrachtet.
            Select Case .datatype
                Case LCTYPE_TEXT:
                    '+++ Datentyp TEXT -> GetStream
                    '+++ employee_text muß vom Typ LCStream sein, sonst Fehler...
                    '+++ Der Textwert wird mit dem lokal installierten Zeichensatz
                    '+++ interpretiert.
                    Set emp_text = .GetStream (1, LCSTREAMFMT_NATIVE)
                    Print .text & " ist " & emp_text.text
```

Abb. 1.61:
Beispielcode für
LCField

```

Case LTYPE_CURRENCY:
'+++ Datentyp CURRENCY -> GetCurrency
'+++ employee_loan muß vom Datentyp Currency sein, sonst Fehler ...
Set emp_loan = .GetCurrency (1)
Print "Das Gehalt des Mitarbeiters beträgt: " & emp_loan.text

'+++ Eine Kopie des ursprünglichen Felds wird angelegt
Set ResultFieldCurrency = ResultField.Copy
'+++ Gehaltserhöhung um 5%
emp_newloan.value = emp_loan.value * 1.05
'+++ Das kopierte Feld wird mit dem neuen Gehalt belegt
Call ResultFieldCurrency.SetCurrency (1, emp_newloan)
Print "Das neue Gehalt des Mitarbeiters beträgt: " & emp_newloan.text

End Select
End With
Next

ODBCConnection.Disconnect

End

MyErrorHandler:
'+++ Hier wird die Fehlerbehandlung ausgeführt.
'+++ Siehe dazu LCSession

End Sub

```

LCStream

Objekte vom Datentyp *LCStream* bestehen aus einem Textstring oder aus einer binären Zeichenfolge mit variabler Länge. Abhängig vom jeweiligen Typ können verschiedene Formate für den Gültigkeitsbereich der Variablen angegeben werden, mit der die Struktur festgelegt wird.

Als Textformate werden verschiedene Zeichengruppen unterstützt, die durch Formate wie *LCSTREAMFMT_NATIVE* für den installierten Zeichensatz auf der lokalen Maschine und *LCSTREAM_UNICODE* für den Default-Zeichensatz des LSX LC ergänzt werden.

Binäre Zeichenfolgen können durch die Angabe von Parametern als unformatierte Objekte (BLOB = Binary Large Object), als Richtext oder als Liste von Text-, Zahlen- und Datumswerten definiert werden.

LCNumeric

LCNumeric ist ein Speicher für Zahlen von sehr hoher Genauigkeit. Per Default-Einstellung werden numerische Variablen dieses Typs mit 88 Ziffern und 44 Nachkommastellen definiert. Die Genauigkeit und die Anzahl der Dezimalstellen können nur bei der Variablendeklaration



definiert werden. Werden bei der Festlegung dieser Parameter Werte benutzt, die außerhalb des Gültigkeitsbereiches für das Objekt *LCNumeric* liegen, resultiert die Ableitung in einem Fehler.

LCCurrency

Die Klasse *LCCurrency* repräsentiert Werte vom Typ Währung mit den gleichen Formatvorgaben und Restriktionen wie der korrespondierende LotusScript-Datentyp *CURRENCY*. Mit einer neunzehnstelligen Genauigkeit und vier Dezimalstellen wird dieser Datentyp verwendet, um eine höhere Genauigkeit als mit Integer- und Floatwerten zu erzielen.

Bei einer Zuweisung von Werten an eine Variable dieses Typs, die jenseits der angegebenen Parameterwerte für Genauigkeit und Präzision liegt, wird ein Fehler ausgegeben und der Variablenwert auf den jeweils maximalen beziehungsweise minimalen Wert gesetzt.

LCDatetime

In einem Objekt dieses Typs werden Datums- und Zeitwerte festgehalten und es werden sowohl Zeitzonen, als auch der Status von Sommer- und Winterzeit gespeichert.

Die Zeitkomponente wird mit einer Genauigkeit von 0,01 Sekunden festgelegt. Da unter Umständen eine Zuweisung eines Zeitwertes an eine Variable dieses Typs erfolgen soll, deren Genauigkeit zum Beispiel höher ist, kann es zu Abweichungen in der Genauigkeit kommen. Aus diesem Grund sollte bei der Variablendeklaration ein Flag (*LCFIELDF_TRUNC_PREC*) gesetzt werden, mit dem dieser Fehler umgangen werden kann.

Durch Setzen weiterer Parameter können die Werte einer Variable vom Typ *LCDatetime* nur aus dem Datumsteil oder nur aus dem Zeitteil bestehen. Wie bei den Variablen vom Typ *LCCurrency* gilt auch bei *LCDatetime*, daß die Variablenwerte im Falle eines Datenüberlaufs auf den maximalen beziehungsweise minimalen Wert gesetzt werden.

1.4.2 Darstellung der Objekte und Methoden

Die in den voran gegangenen Kapiteln beschriebenen Klassen finden sie in den folgenden Abbildungen nochmals in übersichtlicher Form mit den zugehörigen Methoden und Eigenschaften.

Abb. 1.62:
LSX LC –
Objekte und
Methoden (1)





| LCStream | LCDateTime | LCNumeric |
|---|---|---|
| <p>Methoden</p> <ul style="list-style-type: none"> * Append * Clear * Compare * Convert * Copy * DateTimeListGetRange * DateTimeListGetRangeValue * DateTimeListInsertRange * DateTimeListInsertValue * DateTimeListRemoveRange * DateTimeListRemoveValue * Extract * Merge * NumberListGetRange * NumberListGetRangeValue * NumberListInsertRange * NumberListInsertValue * NumberListRemoveRange * NumberListRemoveValue * SortFormat * SortFormat * Substrate * TextListFetch * TextListInsert * TextListRemove * Trim <p>Eigenschaften</p> <ul style="list-style-type: none"> * Flag * Format * Length * MaxLength * RangeCount * Text * Value * ValueCount | <p>Methoden</p> <ul style="list-style-type: none"> * Adjust * Clear * Compare * Copy * GetDiff * SetClearText * SetCharacter <p>Eigenschaften</p> <ul style="list-style-type: none"> * Minute * Second * Seconds6th * Day * Hour * Month * Year * Weekday * Zone * DST * Ticks * Text * Year * Value | <p>Methoden</p> <ul style="list-style-type: none"> + Add + Compare + Copy + Subtract <p>Eigenschaften</p> <ul style="list-style-type: none"> + Precision + Scale + Text + Value |
| | | <p>LCurrency</p> <p>Methoden</p> <ul style="list-style-type: none"> + Add + Compare + Copy + Subtract <p>Eigenschaften</p> <ul style="list-style-type: none"> + Text + Value |

Abb. 1.63:
LSX LC –
Objekte und
Methoden (2)

1.4.3 Einsatzgebiet der LSX LC

Das *LSX LC* ist, wie bereits erwähnt, durch die oben beschriebenen Klassen eine optimale Erweiterung der LotusScript-Entwicklungsumgebung, um auf externe Daten mit Hilfe der zur Verfügung stehenden Konnektoren zuzugreifen.

Durch die Möglichkeit, alle Informationen durch LotusScript weiter zu bearbeiten und unabhängig von ihrem Ursprung zu modifizieren, ist der Einsatzbereich wesentlich größer als bei den konfigurierbaren Methoden LEI und DECS.

Ein erfahrener Lotus-Notes-Entwickler kann, wenn dies die Projekt- und Prozeßvorgaben erlauben, natürlich eine Mixtur aus LEI, DECS und *LSX LC* verwenden. Dabei würden aufwendige und nur mit Hilfe

von LotusScript zu lösenden Probleme durch das *LSX LC* realisiert werden und Datenströme, die sich unter Umständen häufig ändern, durch die Funktionalitäten von LEI oder DECS abgedeckt werden. Dadurch kann, unter der Voraussetzung, daß dies sinnvoll ist, eine Anpassung der Datenströme auch durch weniger erfahrene Entwickler vorgenommen werden.

Für bestimmte Einsatzbereiche sollte besonders geprüft werden, ob alle notwendigen Funktionen und Anforderungen mit dem *LSX LC* gelöst werden können. Teilweise bieten sich hier Speziallösungen, wie das bereits angesprochene *LSX R/3* an, um eine wesentlich komplexere Aufgabenstellung realisieren zu können.

Für alle, denen die konfigurierbaren Systeme wie LEI und DECS nicht genügen, wird über das *LSX LC* eine Script-Erweiterung für Lotus-Domino-Entwickler zur Verfügung gestellt, mit der sich viele Funktionalitäten abbilden lassen. Weiterhin bietet die Nähe zur Programmierumgebung natürlich den entscheidenden Vorteil, daß man Daten direkt vor- und nachbearbeiten kann.

Das *LSX LC* und eventuelle Updates gibt es kostenlos als Download und es läßt sich sehr einfach in die bestehende Entwicklungsumgebung integrieren. In allen Client- und Server-Versionen ab Release 4.63 ist das *LSX LC* allerdings bereits vollständig integriert und braucht nur noch in die Entwicklungsumgebung (LotusScript) eingebunden werden. Die Klassen sind einheitlich für alle angeschlossenen Konnektoren, wodurch sich der Entwicklungsaufwand beispielsweise durch wiederverwendbare Module reduzieren läßt.

Im wesentlichen entspricht die Arbeit mit dem LSX dem Füllen der Datenmasken der konfigurierbaren Methoden. Da aber gerade an dieser Stelle durch die Scriptsprache, bezogen auf die Daten, eine Dynamik eingebaut werden kann, ergeben sich viel flexiblere Anwendungen.

Vor dem Einsatz der LotusScript-Extension sollte genau geprüft werden, in welcher Art und Weise auf die externen Systeme zugegriffen werden soll. Benötigt nur der Server einen Zugang, das heißt, daß alle Zugriffe über den Server abgearbeitet werden, braucht auch nur dort das LSX installiert werden, falls es nicht bereits vorhanden ist. In der Basisinstallation sind auch die Standardkonnektoren enthalten, so daß bei Bedarf nur die *Premium Connectors* nachinstalliert werden müssen. Sollen aber Anwender durch direkte Interaktionen auf die externen Daten zugreifen, ist die Verteilung des LSX auf die entspre-



chenden Workstations ein weiterer Punkt, der beachtet werden muß. Da Lotus Domino hierbei aber eine sehr gute Unterstützung bietet, indem Software, eingebunden in Dokumente, unternehmensweit verteilt werden kann, stellt auch dies kein großes Problem dar.

Obwohl das LSX kostenlos zur Verfügung gestellt wird, sollte aber der Eindruck vermieden werden, daß sich die Anschaffung eines LEI hinsichtlich der höheren Beschaffungskosten nicht lohnt. In der Regel liegt eine individuell programmierte Zugriffsschnittstelle durch den Entwicklungsaufwand in der Summe der Kosten oftmals wesentlich über den Kosten, die durch Kauf, Installation und Konfiguration des LEI entstehen. In dieser Rechnung fehlt natürlich die Hardware, die für einen LEI entweder neu gekauft oder erweitert werden muß. Für den Einsatz von *LSX LC* ist eine Erweiterung der bestehenden Hardwareumgebung nicht zwingend erforderlich.

Literaturliste:

- [Lotus2000] redmond's technology publishing, redmond's inside Lotus Notes/Domino: Schwerpunkt: Domino Enterprise Connection Services, Martin Kuppinger, Ausgabe April 2000, Jahrgang 2, Nr. 4, »<http://www.redmonds.de>«, 2000
- [Lotus99a] Lotus Development Corporation: Lotus Domino R5.0 Enterprise Integration: Architecture und Products, IBM Redbook, »<http://www.redbooks.ibm.com>«, 1999
- [Lotus99b] Lotus Development Corporation: Lotus Domino Release 5.0: A Developer's Handbook, IBM Redbook, »<http://www.redbooks.ibm.com>«, September 1999
- [Lotus99c] Lotus Development Corporation: Enterprise Integrator Domino Connector: LotusScript Extension Guide, »<http://www.redbooks.ibm.com>«, 1999
- [Lotus99d] Lotus Development Corporation: Implementing Lotus Domino Connector and J.D. Edwards' One-World on IBM Netfinity, IBM Redbook, »<http://www.redbooks.ibm.com>«, Juni 1999

- [Lotus99e] Lotus Development Corporation: Lotus Domino Connector for Oracle Applications, Lotus Documentation, »<http://www.lotus.com/home.nsf/welcome/eizone>«, 1999
- [Lotus99f] Lotus Development Corporation: Lotus Domino Connector for PeopleSoft 7 and 7.5, Lotus Documentation, »<http://www.lotus.com/home.nsf/welcome/eizone>«, 1999
- [Lotus99g] Lotus Development Corporation: Lotus Domino Connector for SAP R/3, Release 1.5, User Guide, 1999
- [Lotus98a] Lotus Development Corporation: Domino Enterprise Integration, White Paper, »<http://www.redbooks.ibm.com>«, Mai 1998
- [Lotus98b] Lotus Development Corporation: LotusScript Extension for Lotus Domino Connectors, Reference Guide, »<http://www.redbooks.ibm.com>«, 1998
- [Lotus97] Lotus Development Corporation: Lotus Solutions for the Enterprise, Volume 4, Lotus Notes and the MQSeries Enterprise Integrator, IBM Redbook, »<http://www.redbooks.ibm.com>«, Oktober 1997
- [Haerder90] Härder T., Meyer-Wegener K.: Transaktionssysteme in Workstation/Server-Umgebungen, in: Informatik – Forschung und Entwicklung, S. 127-143, Springer-Verlag, 1990