# Visual DataFlex Roadmap - VDF 17.0

John Tuohy

Data Access Worldwide

# Visual DataFlex 2012 / VDF 17.0

- Visual DataFlex 2012 will have two releases
  - Visual DataFlex 17.0
  - Visual DataFlex 17.1
- I will be showing you what's coming in VDF 17.0
- Stephen will show you what's coming in VDF 17.1
- We have a lot to show, so let's get started with VDF 17.0

# Longer Table and Column Names

- Maximum Table Name (logical name) has been increased from 8 to 31 characters
  - *Employe1* can now be called *EmployeeOvertimeHours*
- Maximum Column names have been increased from 15 to 32 characters
  - *Employee.InsurancePrvdr* can now be *Employee.EmployerInsuranceProvider*
- This applies to all backend databases including the embedded database
- This change is fully backwards compatible
  - No changes required in the table format
  - Names are still stored in Filelist and .fd files
  - If you need compatibility with old versions, just keep the name lengths at their old shorter limit
  - The new size limits were created to maintain maximum backwards compatibility
- This has been at the top of developers wish list for a long time

DATA ACCESS WORLDWIDE

# Studio Enhancements

- Automatic “Todo” support
  - A menu item and a hot key (Ctrl+T) adds a “// Todo:” to your code.
    - You just need to add the todo text
  - A “To Do Items” panel lists all outstanding todo items
    - Clicking on todo item will load that file and jump to that line
    - Todo items will also appear in code-explorer
- Better Multi-File Search support
  - Match whole words
  - Find in files can be accessed from the editor’s context menu
  - It is now easier to add search paths (libraries, etc.)
- There’s more coming, that we can’t tell you about yet

# Video Walkthroughs

- We are working on a series of video walkthroughs and how-tos that will demonstrate a number of the basic features and capabilities of Visual DataFlex
- These will be short and well suited for hosting sites such as You Tube
- We want these to:
  - Show prospective developers the powers of Visual DataFlex and demonstrate how quickly a sophisticated data entry applications can be built
  - Provide these as tutorials for new programmers

# Better Client Web Services

- We've changed the way web-service clients handle those schema types that cannot be mapped to a DataFlex native type or struct
- Previously types that were defined in unusual ways, were mapped to pure XML objects
  - With structs this created a "bubble-up" problem (I'll explain)
  - This required that you deal with the XML, which is more difficult
  - Whenever you see a *Handle* data type, you knew it is going to be more difficult to consume that service
- A new change in how we serialize the XML allows us to isolate these XML islands in much smaller areas
- The end result is that more of the XML parsing and conversion is done for you

# General Entry Improvements

- New Entering / Exiting Messages sent to containers
  - *OnExitArea* and *OnEnterArea*
  - These are sent to the container of the control being entered or exited
  - They are delegated up to all containers within the view/scope
  - They are sent after navigation is complete (a notification, not a verification)

- Improved Masked Date data entry
  - If year is not entered, the current year is used
    - 12/12 is changed to 12/12/2011
  - Works with mm/dd/yy and dd/mm/yy formats
  - No changes required in your application

# Data Dictionary Improvements

- Goals
  - We wanted to make it easier to do some of the things that were confusing or hard
    - There are a number of requirements that are both unique to data entry applications but quite common within these types of applications
    - We wanted to identify these requirements and see if these could be handled more easily and automatically within the DDs
    - We want you to have to do less coding
  - We wanted to provide features and extensions that developers have been asking for
    - We've reviewed your requests and your "how-to" questions to see if we can make changes that makes your life easier.
  - We wanted to look at some of the existing DD features and see if they can be improved
  - We want you to be able to empower your end-user and make it easier to perform data entry and data lookup
  - We want to maintain backwards compatibility with your existing DDs and your existing applications

# DDs: Cascade Delete Validation

- New message: Validate_Cascade_Delete
  - This new validation event is sent to *every* record deleted as part of a cascade delete
  - If an error is raised or the validation returns non-zero the deletion is canceled and the entire transaction is rolled back

```
Function Validate_Cascade_Delete Returns Boolean
    If (OrderDtl.Extended_Price>100) Begin
        Error DFERR_OPERATOR "Cannot delete orders with details items over 100“
        Function_Return True
    End
End_Function
```

# DDs: Pre and Post Find Events

- We've added pre and post Find events

```
Procedure OnPreFind Integer eMessage
Procedure OnPostFind Integer eMessage Boolean bFound
```

- This is sent for DD find (and clear) messages
  - Request_Find, Find, FindByRowId, Find_By_Recnum, Request_Assign and Clear
  - The find type is passed in eMessage
- This is sent to the DD that owns the table for the find
- This is called before / after the DD operation is complete
  - Therefore, it is reentrant safe so you can perform other DD operations

# DDs: Refresh Message sent to DEO Containers

- After a save, find, clear or delete operation DDs now send Refresh to DEO Containers as well as DEO Controls
- A container Refresh does nothing and is ideal for augmentation
  - This provides a hook that allows a view or part of a view to react to the major DD operations
- In addition, we've fixed a bug where DEO expressions did not always receive the Refresh message when it should (it was over optimized)

# DDs: Full DD Support for Text Fields

- Previously
  - With Windows DEOs, the DDs and DEOs only partially supported a DD representation of text fields
    - Text field data was moved directly between the table buffer and the DEO bypassing the DD buffer
    - Most of the time this did not matter (which is why we did this). There are times when this proves to be a limitation.
  - With WebApp and with batch processes, you could use DD text fields but you had to specify in your DDO that you wanted to use extended DD fields
- In 17.0
  - Text DEOs (cDbTextEdit and cDbRichEdit) are now full DEOs and can use the local DD buffers for text fields, just like the other DD fields
    - This is actually configurable for backwards compatibility purposes
  - You no longer need to specify that you want to define an extended field – it's automatic

# DDs: Better Parent / Child constrained Support

- Previously finds in a relates-to constrained child DDO might re-find and refresh the contents of the constraining parent
  - This could result in unsaved changes in a parent DD getting lost
  - This design limitation was particularly apparent with header / detail types of views
  - This is why we always require that headers are saved before entering the detail grid
  - Take a look at the order entry view and you will see custom code that is required to make this all work
  - The framework has worked this way since DataFlex 3.0
- This has been re-engineered for 17.0
  - The DDOs now handle this much more intelligently and saves and finds only occur when you expect them to
  - This means that the header/detail “save first” rule is not needed and that a great deal of custom header/detail view code can be removed
  - This is a bit hard to explain but the net result is that header/detail require less code and they simply work the way you expect them to

# DDs: Null Parent Support

- A child DD may now specify that a parent record may be null
- This is controlled with the new method Set ParentNullAllowed

```
Set ParentNullAllowed SalesP.File_Number to True
```

- The framework now understands the null parent concept and handles it properly
  - No attempt is made to create a new blank record during the save
  - Autofind and Find-Required will accept a null record as valid
  - You can switch a null parent to a real parent and visa versa
- This can be defined at the DD class or object level
- This will be modeled in the Studio

# DDs: DD Remember Field

- DDs have always supported the ability to set default values using field_defaults
  - This has not changed because it's a great feature
- DDs support the Retain / RetainAll field options but these never really fit well within the Framework
  - This has not changed because it's not such a great feature (i.e., left alone compatibility reasons)
- In 17.0 we are introducing a retain option replacement called *DD Remember*
- DD Remember allows you to:
  - Assign and remember a specific value
    - When you clear a view, this remembered value will be used as the new default
  - Remember the last value entered (similar to retain)
    - When you clear a view, the value currently in the DEO will be used as the new default
  - This can be assigned at the DD level or at the DEO level
  - Even better, this can be set directly by your end users
    - Menu items can be assigned to your menus, context menus and tool bars to do this
    - This empowers your users and makes data entry faster and more flexible
    - No programming is required to support this
  - This works differently and properly with parent fields where you can remember an entire parent record
- We will show you this – you really have to see it

# DDs: Committed Records and DD_Commit

- With data entry application a record tends to have two distinct modes.
  - Either it is new or it is committed
    - Most often this aligns with whether a record is new or not (i.e., has it been saved once)
  - A committed record often has restrictions as to what can be changed
    - Once committed, there are certain fields you should not be able to change
    - Once committed, you often want to restrict the switching of certain parent records
  - Enforcing these restrictions requires custom coding – often outside of the DD
- Here is how the DD committed record feature works:
  - You can now specify if a record is committed or not
    - By default, it is committed if has already been saved but you can change this
  - A new DD field option, *DD_Commit*, can be applied to any DD Field. If set, and the record is committed, the DEO for that field will be dynamically disabled
  - A new method, *ParentNoSwitchIfCommitted*, allows you to restrict a committed record's parent from being switched.
  - This takes a common data entry requirement, which can be difficult to implement and automates it.

```
// Once committed, we don't want to be able to change ordered_by
Set Field_Option Field OrderHea.Ordered_By to DD_COMMIT
// If a committed order, don't allow customer parent to be changed
Set ParentNoSwitchIfCommitted Customer.File_Number to True
```

# DDs: Local DD Relationships

- Table relationships can now be defined locally within a DD class or object
- Any DD can now use global relates, which are defined within your table, or the new local relates, which are defined within the DD
  - The pbUseDDRelates property controls this
- If local relates are used you define the relationships with a new message:
  - Set Field_Related_FileField

```
Set pbUseDDRelates to True
Set Field_Related_FileField Field OrderHea.SalesPerson_ID to File_Field SalesP.ID
Set Field_Related_FileField Field OrderHea.Cust_Num to File_Field Customer.Cust_Num
```

- This should reduce the need for alias tables

# DDs: Better Alias Table/DD Support

- It is now much easier to create, maintain and use Alias tables and DDs
- The Studio now has an option for creating an Alias Table & an Alias DD class
- Alias DDs are sub-classed from their master Table's DD
  - The *Set Alias_File* message specifies that the DD sub-class is an alias
- An Alias DD class
  - Inherits the all of the behaviors of the super class except
  - All global relationships and required relationships are cleared for an alias and your are expected to define your own with local relates
  - Prompts and Zooms are cleared for the alias
- There is no need to set alias and master attributes for the tables
  - it is figured out automatically.

# DDs: Better Alias Table/DD Support

- Here is what an alias DD will look like:

```
Use SalesP.dd

Open SalesPManager

Class cSalesPManagerDataDictionary is a SalesP_DataDictionary

    Procedure Construct_Object
        Forward Send Construct_Object
        Set Alias_File to SalesPManager.File_Number
        Set pbForeignReadOnly to True
    End_Procedure

End_Class
```

- This will be modeled in the Studio

# Summary of VDF 17.0

- Longer Table and Column Names
- Studio ToDo panel
- Studio enhanced multi-file search
- Video Walkthroughs
- Easier to Use Client Web Services
- UI: Easier Date entry
- UI: Entering / Exiting delegates to containers
- DD: Cascade Delete Validation
- DD: Pre and Post Find Events
- DD: Better Refresh Event
- DD: Full DD support for text field DEOs
- DD: Better handling of header/detail views
- DD: Null Parent Support
- DD: DD Remember
- DD: Committed Record feature
- DD: Local DD Relationships
- DD: Better Alias Table Support