Spreadsheet Compatibility

Backwards, Forwards, and Sideways

Jody Goldberg
jody@novell.com
OooCon 2006
Lyon, France

September 19

Novell®

Spreadsheet Compatibility

Interesting

- Calculation
  - Reproducability
  - Provide clear warnings when results **may** differ

Uninteresting (for this talk)

- Visual fidelity
- User Interface
- Organizational Tools
- Scripting

Why is this Interesting

Blocks migration

- User's usually have existing data

Increases error rates

- Error rates are relatively constant at 5% of cells
- Complex or unfamiliar systems will triple that

Why People Will Pay

"Spreadsheets should be treated as software applications in their own right."

Price Waterhouse Regulatory Guide

Financial Regulators

SOX 302 & 404

Pharmaceutical Regulators

Title 21 part 11

Types of incompatibility

Easy

- Bounds
  - Floating Point
  - Grid Size
- Versioning
  - Application
  - 3rd Party extensions

Hard

- Workbook Functions
- Formula Interpreter

Floating Point

Precision

- single
- double
- quad

CPU

- Between Vendors
  - Intel vs AMD
- Different Modes
  - ARM 'Vector Floating Point' (hard & soft)
  - ARM 'Floating Point Accelerator' (hard & soft)

Grid Size

Name Wrapping

- Named range with a **relative** reference

  eg 'A1' created in B1 and evaluated in A2

Full Row/Column references

=COUNTBLANK(B:B)

A1 style addresses conflict with useful names

- Column 1000, Row 1 = ALL1
- Column 8509, Row 2 = LOG2
- Column 364239 = TRUE
  - A non-issue for ODF ⅓ because it is smart enough to mark references
    - [.LOG2]
  - ⅓ because ODF does not support full column refs yet
  - ⅓ because ODF does no support literal booleans fully yet

Application Versioning

Most formats contain version tag

- Most are limited to integer versioning
- Some also differentiate from a family of implementations

No central registry

- Simple with a single application to know what changed between versions
- Almost impossible to know what change for other implementations or architectures

Dangerous

- Enough version info provides an attack vector

Addin Versioning

A common idiom

- Do not trust functions in spreadsheet
- Use calc as deployment platform for custom analytics

Stored Values

- Allow external tools to check generating addin

Solutions for Easy Problems

Goals and Tools

Goals

- Do not need to **solve** the problems
- Know when something might change
  - avoid false positives
- Provide reasonable warnings of where to look for changes

Tools

- Use existing file format
  - Adapt existing tags when possible
  - No format provides the full set of detail
- Add additional data
  - Requires standards approval in some cases, impossible in others
  - Keep set to a minimum.

XML Formats

ODF and XLSX

- XML based
  - 3rd party tools are likely to parse the results
- Easy to Extend (With standards approval)

```
<characteristics>
    <bounds>
        <fp precision=”quad”/>
        <grid max-col=”256” max-row=”65535”/>
    <bounds/>
    <addins>
        <addin name=”foo” version=”id”/>
    <addins/>
<characteristics/>
```

- Fall back on known current details if unspecified
- Can get extensions into MS Office Open filters

Binary Formats

XLS

- Warn about bounds when generating
  - binary format already has hard coded maxima
- Few external tools parse format
- Can add additional records
  - Only OOo (and other OpenSource spreadsheets) would recognize them
  - MS Excel ignores the unknown records

Add 'OOOGENERATED' (0x8D0)

- an empty biff record to differentiate xls generated by OOo and MS XL.

Add 'OOOAddinVersion' (0x8D1)

- similar to xml record

Types of Compatibility

Backwards

- Never change existing behavior
- Even if it's broken

Forwards

- Ensure that old apps can round trip new constructs

Sideways

- Give other implementations enough detail to degrade gracefully.
- Even across file formats.

Function Incompatibilities

Name Conflicts

- LOG vs LOG10

Parameter Conflicts

- Different signatures
  - Adding R1C1 support to ADDRESS and INDIRECT
    Current xls importer strips the unsupported arguments and restores them on export
- Extending a function
  - Adding a parameter
  - Extending the allowable value for a parameter

Behavior differences

- Improving Accuracy
- Type Conversion

Why is this difficult ?

Just create a new function for everything

- FOO_v2
- What could be simpler ?
- Backwards becomes simple
  - Ignoring cost of maintaing multiple copies of functions in-perpetuity

Forward Compatibility

- Need fallback mechanism old version does not know about new name.

Isn't it all being Standardized

- ODF formula subcommittee (previously OpenFormula)
- OfficeOpen includes partial spec

Don't Forget Sideways

Can we use file format as a flag ?

- When loading an xls use XL_FOO
- No :-(

Trans-format Roundtripping

- OOo(xls)
- XL(xls) -> OOo(xls) -> XL(xls)
- OOo(xls) -> XL(xls) -> OOo(xls)

Available Tools

ODF

- Generator version
- A namespace for each expression
  - “oooc:” The existing name space for 2.0 -> 2.0.3
  - “of:” 'OpenFormula' namespace now supported in 2.0.4

OfficeOpen (ECMA TC45)

- Version of appliction
- 'Platform' of application
- A plethora of random backwards compatibility flags

Proposed Solution

Add a function Map for behavior changes

- Use 'default' names in exported formulas
  - Ensures backwards and forwards compat
  - Newer versions can lookup expected version
  - Older apps can warn of mismatches in impacted cells
- Use versioned names for signature changes
  - Too large a change to allow old apps without forcing old apps to know about the map

Summary

ODF Extensions

- Function map to completement the baseline versions
- Minor additional characteristics tree with details

XLS Extensions

- Add 2 new records for characteristics


Novell®