Managing Autodesk® Revit® links with the 2013 Revit API

Arnošt Löbel
Sr. Principal Software Engineer

Diane Christoforo
Sr. Software Engineer

Autodesk University

Class Summary

This class will outline the scenarios and explain common pitfalls of working with links via the Autodesk Revit API.

We are going to:

1. Detail the creation of Autodesk Revit link types and instances
2. Recommend how to easily check or modify the parameters of links
3. Walk you through material that describes ways to inquire data about the various types of external files, even in closed Revit documents

Learning Objectives

At the end of this class, you will be able to:

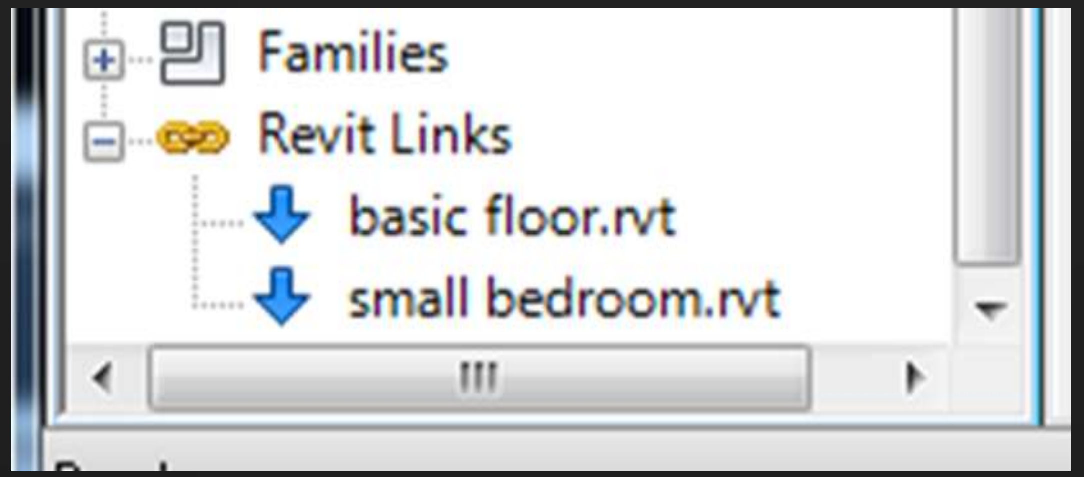
1. Create link types and instances (new in 2013)
2. Query link parameters and properties
3. Examine link path information with ExternalFileReference
4. Modify links using TransmissionData (in closed Revit files)
5. Use *eTransmit*, a downloadable add-in for Autodesk® Revit®
6. Write your own eTransmit application to suite your exact needs

Getting familiar with the jargon

Element classes

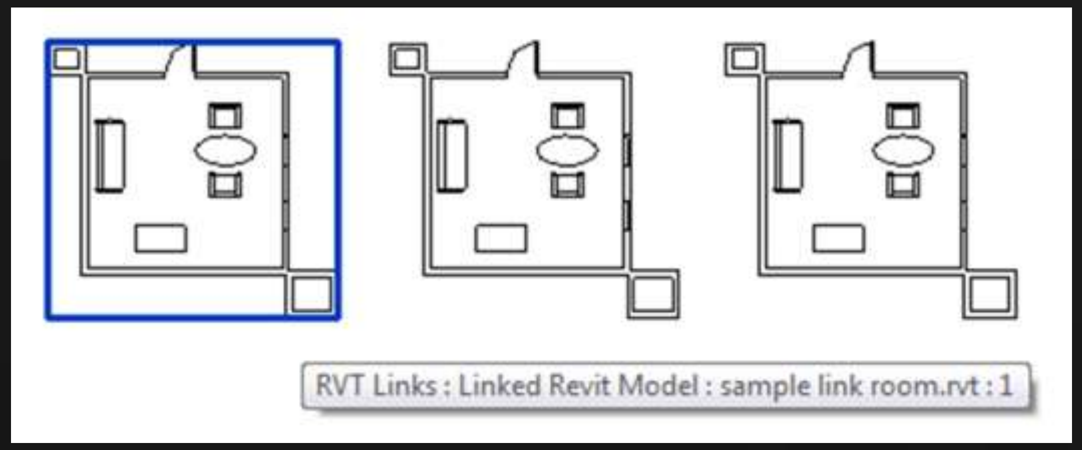
RevitLinkType

- One for each and every linked document



RevitLinkInstance

- Instances of respective Link Types



Non-Element classes

ExternalFileReference

- Stores path information about a linked Revit file, CAD link, and few other types including the keynote table.

TransmissionData

- Stores a set of objects of type ExternalFileReference. Can be read and modified in a closed document!

Part1
Creating Revit Links

Attaching a link file to a document

RevitLinkType.Create – a static method

Arguments:

1. Document – the hosting document
2. ModelPath – fully qualified path (file or server) to the link file
3. RevitLinkOptions – controls if link paths to be stored relatively of absolutely

Returns: RevitLinkLoadResult

- Contains result status (which could identify an error)
- Also contains an ElementId of the new Link Type (in case of a success)

Happy note: 2013 bug fixed in UR. Links now stay loaded after document is reopened.

Creating an instance of a link

RevitLinkInstance.Create – a static method

Arguments:

1. Document – the hosting document
2. ElementId – Id of a RevitLinkType.

The Link Type must be already loaded In the host document!

Returns: RevitLinkInstance

Instances are placed origin-to-origin!

- You can move the link instance by modifying its Location property
- Note: Unlike with the UI, you cannot align the link using shared coordinates
  You can access the Project Location of the host project, but you will not
  be able to access the linked document to look up its Project Location.

Possible results of loading link types

Result	Meaning
LinkLoaded	Success ⇨ the Element Id is of a valid link type
LinkNotFound	Most likely there'll be an exception up front
LinkNotOpenable	Revit tried to load the file but failed; probably a corrupted file
LinkOpenAsHost	The file is already opened directly. It is not allowed to have a file open both as a host and link at the same time
SameModelAsHost	Trying to link the host model into itself
SameCentralModelAsHost	Trying to link a local model into its central model or vice versa
LinkNotLoadedOtherError	Something unexpected occurred (not common)

Demo 1a

Creating link types and instances

RevitLinkType – available methods

GetChildIds – element Ids of immediate children link linked this link document

- If A ⇨ B ⇨ C, then B is the only child of A

GetParentId – element Id of the Link hosting this link document

- If A ⇨ B ⇨ C, then B is the parent of C, as A is of B, but `InvalidElementId` is returned for A

GetRootId – element Id of the very root Revit link effectively hosting this link doc

- If A ⇨ B ⇨ C, then A is the root for both B and C, but `InvalidElementId` is returned for A

static GetTopLevelLink (Document, ModelPath) – top link or `InvalidElementId`

static IsLoaded (Document, ElementId) – is this link type currently in memory?

Related Parameters and Properties

RevitLinkType

1. Parameter “*Room Bounding*” [WALL_ATTR_ROOM_BOUNDING]
2. Read-only Property AttachmentType
   a) Overlay – only shown in its host
   b) Attachment – brought along to the host’s host (Not supported to be set via by API!)
3. Read-only property IsNestedLink – whether it is top-level or nested child

RevitLinkInstance

1. Parameter “*Name*” [RVT_LINK_INSTANCE_NAME]
   - It is not the path. It’s the name assigned to the link. Can be modified. (A number by default)
2. Property Location
   - Location of the instance. Can be modified.

Demo 1b

Modifying link types and instances

Part 2
External File Reference

Types with External File References

1. Revit links [`RevitLinkType`]
2. CAD and DWF files [`CADLinkTYpe`]
3. Rendering decals [`Element`] – not directly exposed
4. Keynote table [`Element`] - not directly exposed

Notes:

- Only type elements have file references, not instances
- But not every external file has an External File Reference associated (e.g. Point clouds, Materials, MEP lookup tables, etc. don’t)

Accessing External References

A. If you already know or have the element of a link type

1. Element.GetExternalFileReference (Document)
   - This element would return true to IsExternalTypeReference
2. ExternalFileUtils.GetExternalFileReference (Document, ElementId)

B. If you want all file references in a document

1. ExternalFileUtils.GetAllExternalFileReferences (Document)

   Note: Returns references of all types (with references), not just Revit links.

Public Methods and properties of EFR

ExternalFileReferenceType – RevitLink, CADLink, Decal, Keynote

GetLinkedFileStatus – Loaded, Unloaded, NotFound (*generally, unavailable*)

PathType – Relative, Absolute, RevitServer, Content

GetPath – path to the link file (as it was given)

GetAbsolutePath – absolute path to the link file

GetReferencingId – the element associated with the external reference

Once loaded, External References <u>are immutable</u>, thus all the properties are read-only

A closer look at Path Type

There are 4 different path types. Not all of them can be used with just any link though.

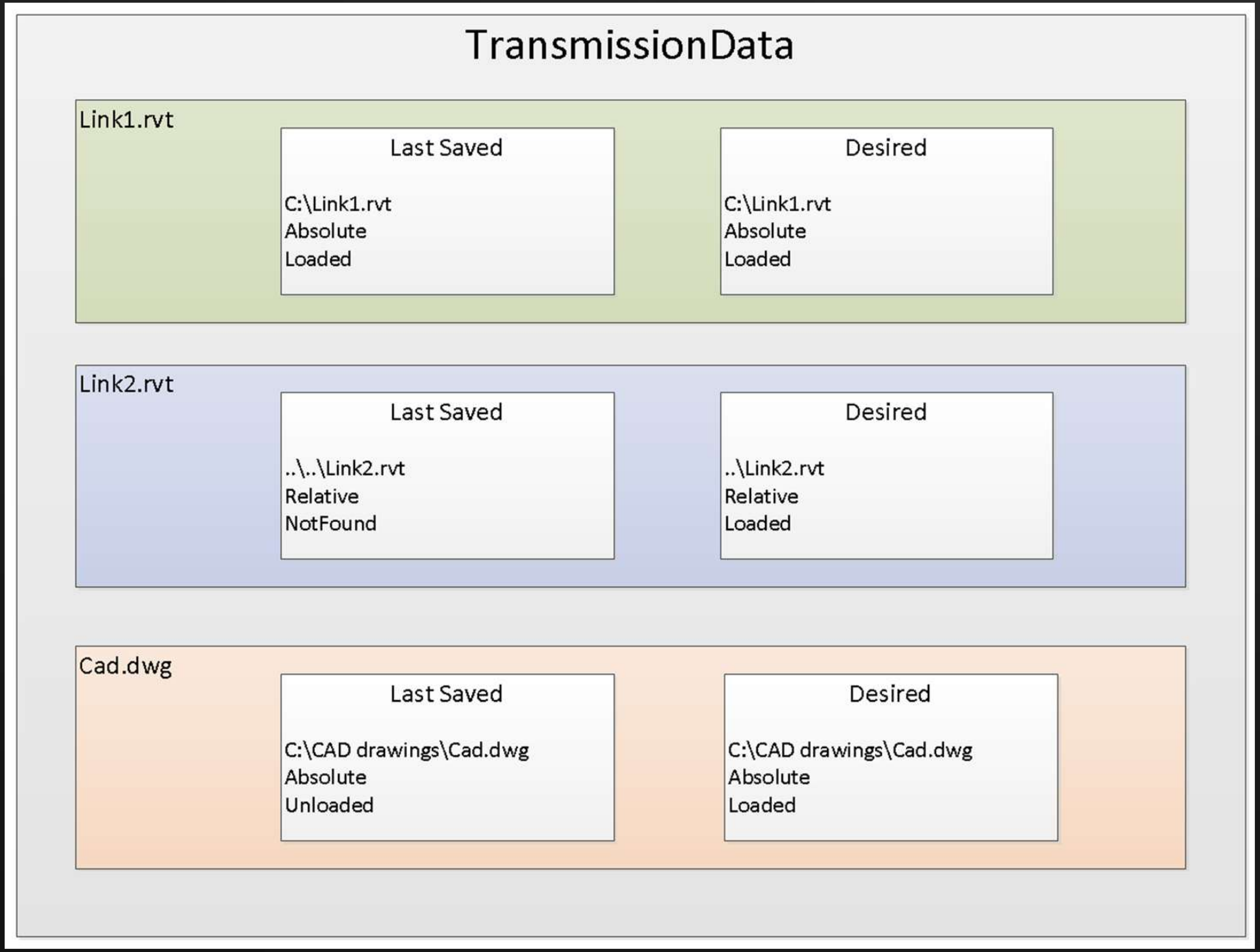
Absolute	Can always be used with any reference file as long as the file physically exists at that location.
Relative	Can be used for any file as long as the relation is valid.
Content	Path relative to the “*Data”* folder in Revit. Valid for <u>keynotes</u> and <u>decals</u> only.
RevitServer	Path on a Revit Server. Can be used for Revit links <u>only</u>, providing the local client has access to that server.

Use ExternalFileReference.IsValidPathTypeForExternalFileReference (*Path Type*) to test whether a Path Type would be valid for a particular file reference.

Demo 2
Browsing External Type References

Part 3
Transmission data

Transmission Data Overview



TransmissionData stores a collection of ExternalFileReferences. For each link in the Revit model, TransmissionData stores two External Type References:

Last Saved - stores the state of the link as of the last time the Revit model was saved to disk.

Desired - reflects what Revit will try to do with the link the next time the model is opened

Note: Both EFRs are always available.

Essentials about Transmission Data

TransmissionData was created to support eTransmit for Autodesk Revit

- TransmissionData record is saved in a separate data-stream with a document
- That allows read and write access without having to open the entire file in Revit

Revit reads the TransmissionData only from files marked as “*transmitted*”

- It is because to read the TransmissionData, work-shared files must first be detached from central
- To mark a file as transmitted, set the IsTransmitted property of the TransmissionData to “true”

TransmissionData record can be modified only in a closed document

TransmissionData record cannot be used to add or remove links

Some TransmissionData facts

1. You cannot unload rendering decals or the keynote table, so you will need to filter those types out if unloading all links.
2. TransmissionData doesn’t report nested Autodesk Revit links.
   - If all links of all levels are to be transferred, their respective TransmisionData records must be modified individually.
3. The flag *IsTransmitted* must be set to *true* before calling WriteTransmissionData

Steps to modify TransmissionData record

1. Obtain a FilePath of closed Revit document
2. Read TransmissionData record from the document
3. Obtain Element Ids of all external file references stored in the data record
4. For each link you want to change, set the desired reference data
5. Set the TransmissionData's IsTransmitted property to *true*
6. Write the modified TransmissionData record back to the document
7. Document can now be reopened with relocated links

The steps as (pseudo-)Code

```
1. FilePath fpath = new FilePath("c:\\temp\\myfile.rvt");  // absolute path!

2. TransmissionData tdata = TransmissionData.ReadTransmissionData(fpath));

3. Ilist<ElementId> refelems = tdata. tData.GetAllExternalFileReferenceIds();

4. foreach (ElementId id = refelems)

   ExternalFileReference efr = tdata.GetLastSavedReferenceData(id);

   A. // Keeping the path, but changing the load state
      tdata.SetDesiredReferenceData(id, efr.GetPath(), efr.PathType, false);

   B. // Chaning the link to a relative file, and setting it to be loaded
      tdata.SetDesiredReferenceData(id, "link.rvt", PathType.Relative, true);

5. tdata.IsTransmitted = true;  // make sure the file is mark as transmitted!

6. TransmissionData.WriteTransmissionData(fpath, tdata)); // write the data back
```

Public Methods and properties

static ReadTransmissionData (ModelPath)

static WriteTransmissionData (ModelPath, TransmissionData)

static IsDocumentTransmitted (ModelPath)

GetAllExternalFileReferences ()

GetLastSavedReferenceData (ElementId)

GetDesiredReferenceData (ElementId)

SetDesiredReferenceData (ElementId, ModelPath, PathType, bool /*should load*/)

IsTransmitted – A flag marking a revit file as transmited.
Revit ignores the record unless the document is marked as transmitted!

UserData – custom string to attach. Revit does not use it.

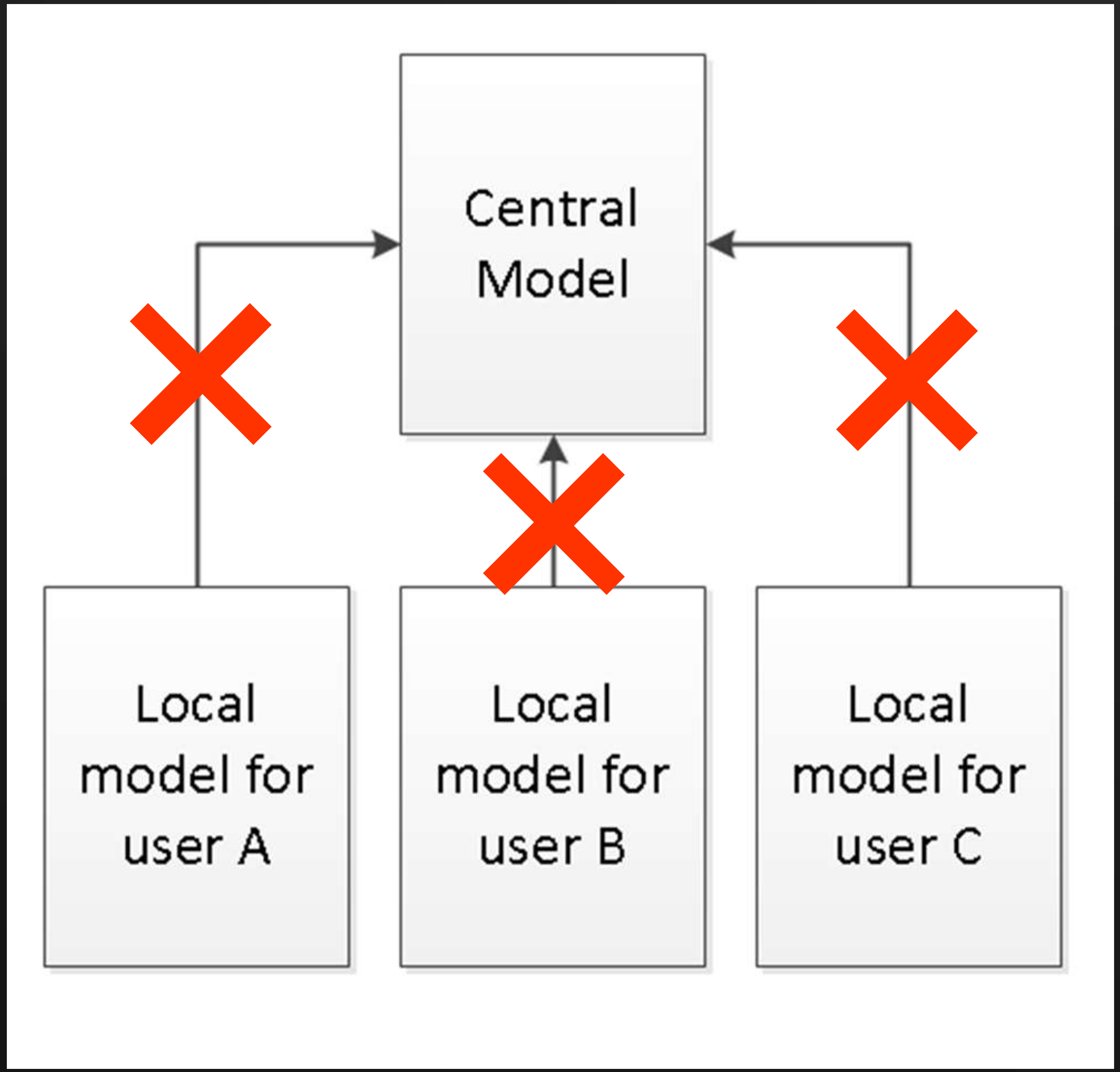
Demo 3
Working with Transmission Data

Part 4
Implementation Notes

Limitations of Autodesk Revit 2013 Links API

1. Cannot reload or unload links while the host document is opened.
2. Cannot create nested links from scratch
3. Cannot link in models using “By Shared Coordinates”
4. API clients cannot (directly) access the linked document

“Attached” vs. “Detached” from Central”?



In a work-shared project, there is a “central” model which is the master version of the project. Each user has their own “local” model, which is kept synchronized with the central copy.

When a model is “Detached from Central”, it becomes an independent model. Users in local models can no longer use “Synchronize with Central” to send changes to the central model.

They also cannot use “Reload Latest”

to get changes from the central.

TransmissionData – Detaching from Central

Opening a transmitted work-shared model will make it “detached”

1. Local model will become incompatible with the central model
2. Central model will invalidate all of its local models

Changing TransmissionData in a work-shared model

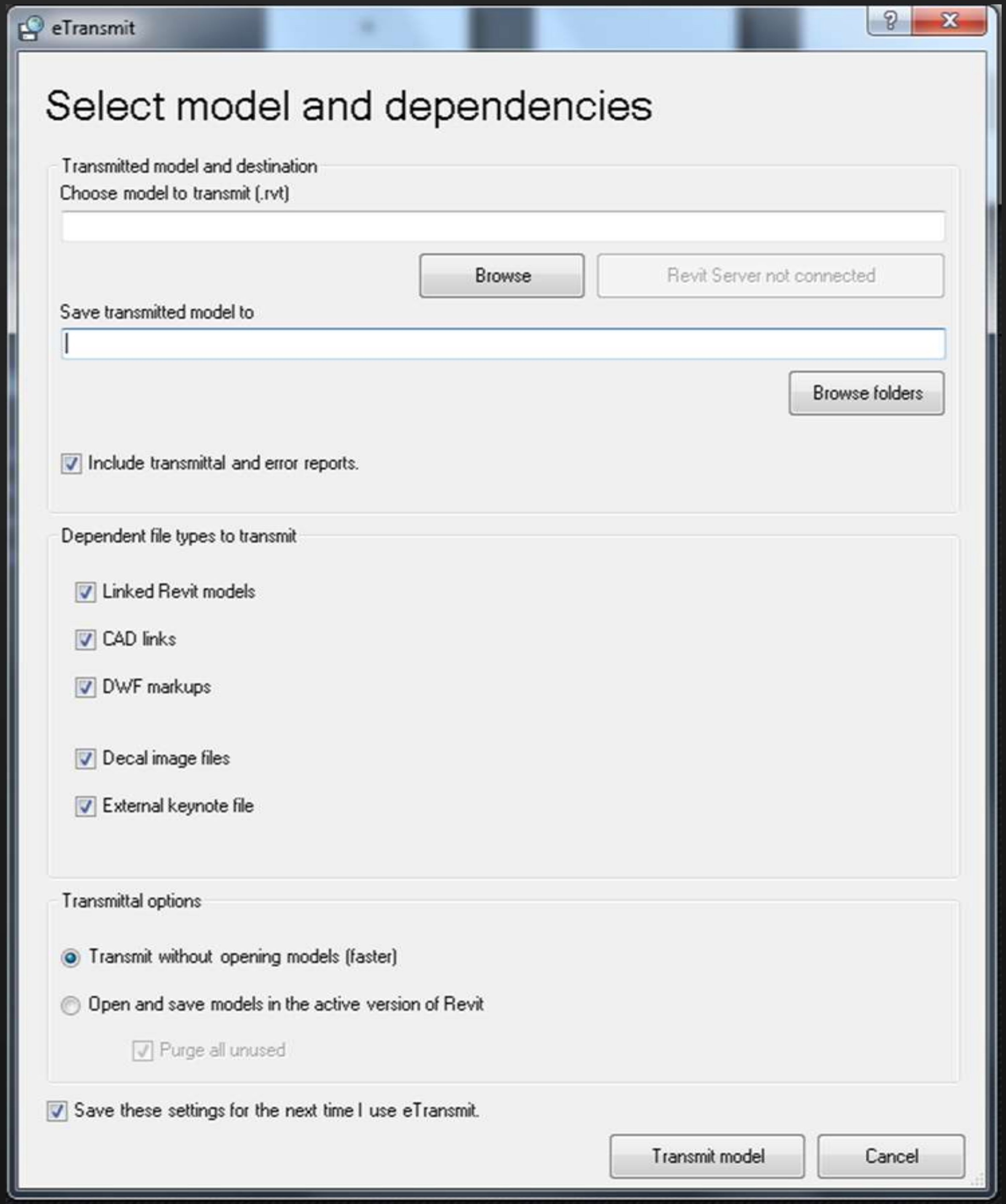
1. Every local user saves to central.
2. Modify the central model. Open and save it, then mark it as no longer transmitted.
3. Make new locals from the central model.

Part 5
eTransmit

eTransmit for Autodesk Revit

It's an add-in that allows bundling up an entire model as a package, so it can be sent and open in at another location.

eTransmit for Autodesk® Revit® 2013 and 2012 is available on subscription site subscription.autodesk.com



Select model and dependencies

Transmitted model and destination

Choose model to transmit (.rvt)

Save transmitted model to

[✓] Include transmittal and error reports.

Dependent file types to transmit

[✓] Linked Revit models

[✓] CAD links

[✓] DWF markups

[✓] Decal image files

[✓] External keynote file

Transmittal options

(●) Transmit without opening models (faster)

( ) Open and save models in the active version of Revit

[✓] Purge all unused

[✓] Save these settings for the next time I use eTransmit.

How *eTransmit* works

1. Choose a Revit model
2. eTransmit will find **all** of that model’s links, recursively
3. All the link files will be copied to a chosen folder
4. Transmission Data records will be re-directed (relatively to the host)
5. Result is a folder that can be sent anywhere knowing all links will be found without getting annoyed with using *Reload From*

Demo 4
Add-In Presentation

Questions?

Autodesk University