Asynchronous Download

QA-IQ

- Data download
- Monikers
- Binding
- URL monikers
- ATL support

315

Data download

QA-IQ

- Controls may have large data files
  - Audio, video
  - Database recordsets
- Need to locate data and download it
  - Process known as binding
- Download needs to be asynchronous
  - To avoid locking the UI
  - To allow progressive rendering

316

Uniform Resource Locators (1)

QA-IQ

- **WWW uses Uniform Resource Locators (URLs) to encode names and addresses of resources on the Internet**
- **URL names exist within a namespace**
  - **Each namespace is uniquely identified**
  - **Defines name format**
- **URL syntax has format <namespace>:<name>.**
  - **e.g. http://www.qatraining.com/index.html**
    - <namespace>=http
    - <name>= //www.qatraining.com/index.html

317

Uniform Resource Locators (2)

QA-IQ

- **Extensible syntax**
  - **Can encode any naming scheme**
- **Consumers of a name need have no explicit knowledge of the namespace**
  - **There is a single binding process**
- **COM moniker architecture offers a convenient programming model for working with URLs**

318

Monikers(1)

QA-IQ

- Objects, data and resources need to be named
  - Often names need to be persistently stored
  - Names implicitly refer to location
- Name format and meaning depend on context
  - Different types of name for file systems, databases, WWW etc
  - Different location logic also
- Consumers can't continually be extended to cope with different types of names
  - Need a single naming and binding model
- Need a namespace

319

Monikers(2)

QA-IQ

- **A COM object that encapsulates a name**
  - **Each moniker class is a namespace handler**
  - **e.g. file monikers, URL monikers**
  - **Each moniker class identified by CLSID**
- **Implements IMoniker**
  - **Only way of manipulating a moniker**
  - **Polymorphic binding of name to object**
- **Name consists of native name data plus display name**

320

System Moniker Classes

QA-IQ

- File Moniker
  - Encapsulates UNC file names, file association yields object
- URL Moniker
  - Encapsulates URLs
  - ftp, http, gopher and potentially others
- Item Moniker
  - Allows a name to be more specific, e.g. item within a file
  - Must be part of a composite to define context and is resolved by some object 'container' with IOleItemContainer
- Pointer Moniker
  - For already running objects
  - So that running and passive objects can be treated the same
- Composite Moniker
  - Arbitrary name as a collection of monikers
  - Knows how to manage relationship between composite parts

321

The COM system provides a set of standard moniker classes. Each has its own class ID.

File Monikers encapsulate UNC filenames. Construct them with CreateFileMoniker("filename",...). The process of binding will find a CLSID associated with the file, instantiate it and pass it the filename through IPersistFile::Load().

The WWW uses Uniform Resource Locators (URLs) to encode the names and addresses of objects on the Internet. URL monikers encapsulate certain URLs. Since URLs frequently refer to resources across high-latency networks, the URL moniker supports asynchronous as well as synchronous binding.

A pointer moniker is a handy way of encapsulating an already instantiated and running object. Create them with CreatePointerMoniker(pUnknown,...). Binding a pointer moniker just calls QueryInterface() on the pointer for the desired interface. Pointer monikers allow us to name and bind to passive and active objects in the same way. See the slide on the Running Object Table later.

Both file and pointer monikers can be used on their own, i.e. not as part of a composite.

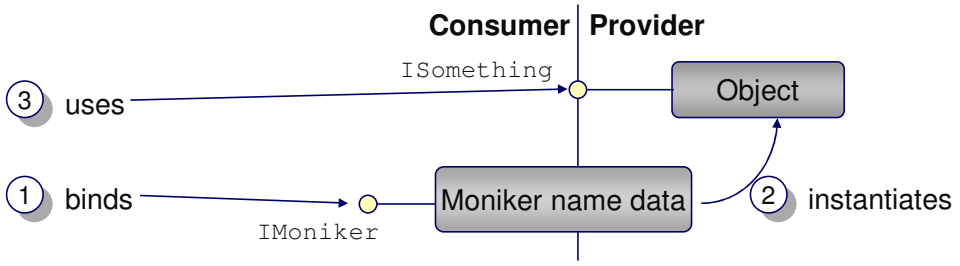
An item moniker however can only be used as part of a composite, as it relies on the moniker to its left for its context. For instance, in the composite moniker "c:\temp\test.xls!Sheet1!R1C!:R2C2", the item moniker identifying row1,col1 to row2,col2 only makes sense in the context of a particular worksheet, in this case, Sheet1. Likewise the item moniker identifying the worksheet, Sheet1, only makes sense in the context of a workbook, in this case c:\temp\test.xls. The workbook is represented by a file moniker which needs no further context. Item monikers can be strung together arbitrarily to form very complex and specific names.

Whilst the meaning of the name data in file and pointer monikers are understood by those moniker classes and can be bound independently, the meaning of an item moniker is decided by the component that gave it out and is resolved by its implementation of IOleItemContainer.

Binding of Monikers

QA-IQ

- **Consumer calls IMoniker::BindToObject() or IMoniker::BindToStorage()**
  - **Asking for interface required to manipulate object**
- **Moniker uses name data to locate, instantiate and initialize object**
- **Moniker passes requested interface pointer back to consumer**
  - **Moniker drops out of the picture**
- **N.B. moniker is a different COM object than the one it refers to!**



322

Once a consumer of a moniker has an IMoniker then the binding process is polymorphic, regardless of what kind of name the moniker represents.

The consumer simply calls the IMoniker::BindToObject() specifying the interface required on the object to be instantiated. Once this call completes successfully then the consumer has an interface on the named object and the moniker drops out of the picture.

Note that a moniker is a COM object in its own right and is not the same as the COM object to which it refers. Once a moniker has been used to bind to an object, it leaves you free to manipulate the object directly.

There are some other details to the binding process, such as how long to wait for a successful bind or performing an asynchronous bind, and the question of bind contexts, but these will be addressed later.

The consumer could also call IMoniker::BindToStorage() which retrieves an interface pointer to the storage that contains the object identified by the moniker. Unlike IMoniker::BindToObject(), this does not activate the object identified by the moniker.

Composite Monikers

QA-IQ

- Collections of monikers that can be used to represent arbitrarily complex names
  - It knows how to manage each through its IMoniker interface in order to compose, bind, enumerate etc
  - It may be able to provide more knowledgeable handling based on the CLSIDs of the contained monikers
- Composite monikers can contain composite monikers
- COM provides a generic composite moniker as a standard class
- Doesn't know anything detailed about contained monikers
  - Except how to compose, bind, enumerate etc

323

Binding a Composite Moniker

QA-IQ

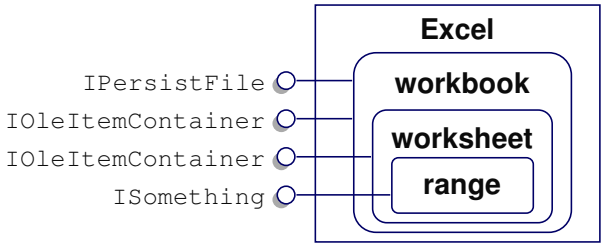
- Consumer will call BindToObject() on outer composite IMoniker, asking for an interface
- What a composite bind does
  - If name references object active in ROT, get it and QueryInterface() for requested interface
  - Separate composite into right moniker and leftmost moniker
  - Return result of call to BindToObject() on right moniker passing leftmost moniker as a parameter
- If leftmost moniker is composite then we get recursive! This stops when leftmost moniker
  - Is in ROT
  - Doesn't depend on anything to the left (it may or may not be the first moniker in the composite). Only file monikers, URL monikers or a suitable custom moniker bind in this way

324

Binding a File/Item Composite

QA-IQ

- e.g. c:\temp\test.xls!Sheet1!R1C1:R2C2
- The file moniker and all but one item moniker refer to container objects that must support IOleItemContainer to resolve names of contained objects
  - Each item moniker requires a context in which to bind
  - Range can't exist without worksheet and it's definition is relative to a worksheet
- The file moniker refers to an object that must also support IPersistFile
  - To tell it which file



325

Let's call the outer composite moniker the workbook/worksheet/range composite moniker. The consumer of the workbook/worksheet/range composite moniker calls its IMoniker::BindToObject() specifying ISomething as the interface required to manipulate the resulting range object.

The workbook/worksheet/range composite moniker will bind right to left. In order to satisfy the bind, it splits its contents into two monikers, the leftmost inner composite moniker (call it the the workbook/worksheet composite moniker) and the last item moniker (call it the the range item moniker). It then delegates to the range item monikers IMoniker::BindToObject(), passing the workbook/worksheet composite moniker as a parameter and specifying ISomething as the required interface. The ISomething interface pointer will be passed back to the consumer as a result of the bind.

Problem! The range item moniker has a name that it can't make any sense of. So it has the ask the workbook/worksheet composite moniker to resolve it. It uses the workbook/worksheet composite monikers IMoniker::BindToObject() to request IOleItemContainer and uses IOleItemContainer::GetObject() to retrieve the range object, specifying the name of the range and ISomething as the required interface, which it will pass back to the workbook/worksheet/range composite moniker as a result of the bind.

The workbook/worksheet composite moniker will bind right to left. In order to satisfy the bind, it splits its contents into two monikers, the leftmost file moniker (call it the the workbook file moniker) and the last item moniker (call it the the worksheet item moniker). It then delegates to the worksheet item monikers IMoniker::BindToObject(), passing the workbook file moniker as a parameter and specifying IOleItemContainer as the required interface. The IOleItemContainer interface pointer will be passed back to the range item moniker as a result of the bind.

Problem! The worksheet item moniker has a name that it can't make any sense of. So it has the ask the workbook file moniker to resolve it. It uses the workbook file monikers IMoniker::BindToObject() to request IOleItemContainer and uses IOleItemContainer::GetObject() to retrieve the worksheet object, specifying the name of the worksheet and IOleItemContainer as the required interface, which it will pass back to the workbook/worksheet composite moniker as a result of the bind.

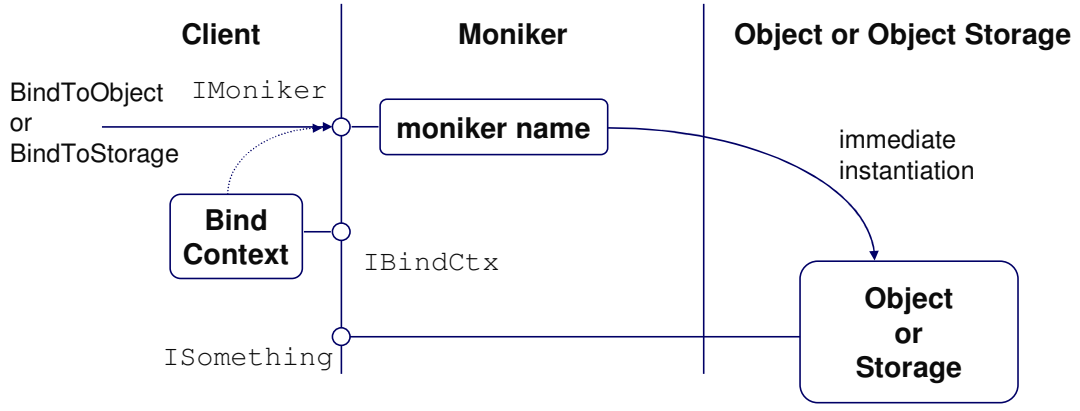
The workbook file moniker will use the CLSID associated with the file to instantiate the workbook object. It queries the workbook object for its IPersistFile and calls IPersistFile::Load() passing the workbook filename represented by the workbook file moniker as a parameter. This tells the workbook object which workbook it is dealing with. It then queries the workbook object for its IOleItemContainer which it will pass back to the worksheet item moniker as a result of the bind.

And so everything filters back and the process is complete.

Synchronous Binding

QA-IQ

- **IMoniker::BindToXXX() returns synchronously with the object/storage or an error**
- **Client can specify timeout period in bind context**
  - **Monikers should take account of this**



327

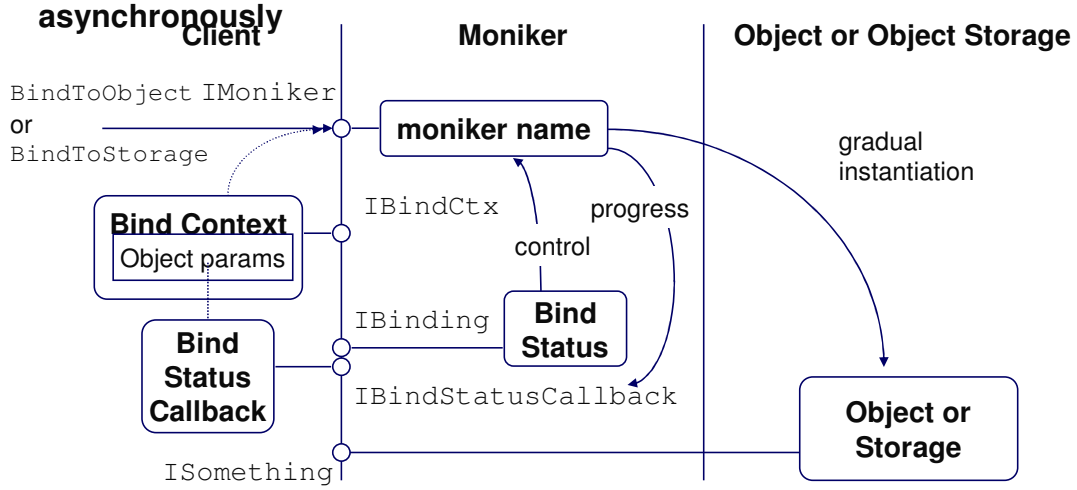
With a synchronous moniker bind then the call to IMoniker::BindToObject() or IMoniker::BindToStorage() will not complete until either the required interface pointer on the object/storage is returned or an error occurs.

Whilst it is possible for the bind client to specify a maximum time period within which the bind must complete, there is no way for the client to to interrupt the operation, suspend/resume the operation or provide additional information during the bind. The client would also have to spawn another thread if it wanted asynchronous binding with regard to the binding thread.

Asynchronous Binding

QA-IQ

- **IBindStatusCallback registered with bind context**
- **IMoniker::BindToXXX() returns immediately**
- **IBindStatusCallback gets bind progress and object/data asynchronously**



328

IBindStatusCallback

QA-IQ

- **Async monikers use this to inform clients of an async moniker bind of bind progress**
- **Client implements a bind sink callback**
  - **Receives async bind info**
  - **Is a private COM object**
  - **Exposes IUnknown and IBindStatusCallback**

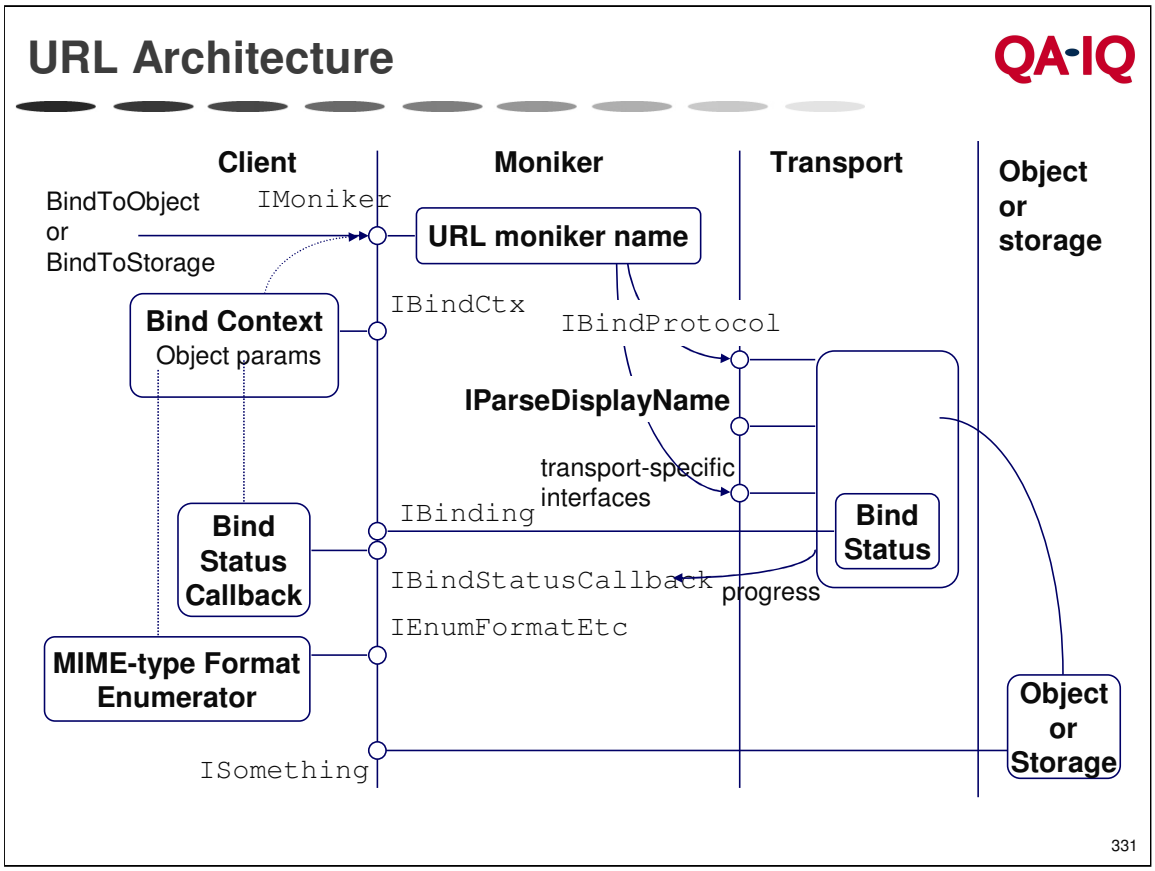
329

URL Monikers

QA-IQ

- Provide namespace handlers for URLs
  - Naming and binding
- Support asynchronous binding
- Support multiple protocols
  - ftp
  - http
  - gopher
- Extensible

330


URL Architecture
QA-IQ
Client
Moniker
Transport
Object or storage
BindToObject or BindToStorage
IMoniker
URL moniker name
Bind Context
Object params
IBindCtx
IBindProtocol
IParseDisplayName
transport-specific interfaces
IBinding
Bind Status
Bind Status Callback
IBindStatusCallback
progress
IEnumFormatEtc
MIME-type Format Enumerator
Object or Storage
ISomething
331

Creating and Using a URL Moniker

QA-IQ

- CreateURLMoniker() to yield an IMoniker
  - <protocol>:<name> format
- Supported protocols
  - ftp
  - gopher
  - http
- Bind in the normal way with IMoniker::BindToObject(), IMoniker::BindToStorage()
  - Synchronously
  - Asynchronously
- GetClassURL() returns the CLSID associated with the resource specified by URL

332

ATL Support for Monikers

QA-IQ

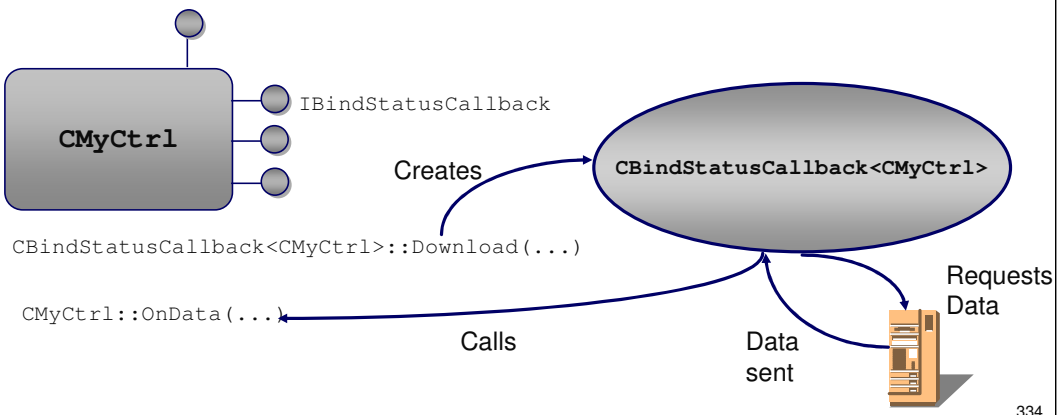
- No moniker class
- Support for asynchronous download
  - Through CBindStatusCallback<>
- Control must derive from
  - IBindStatusCallbackImpl<>

333

Async download in ATL

QA-IQ

- Use CBindStatusCallback<>::Download()
  - Static member function
- Implement OnData()
  - Called by CBindStatusCallback when data arrives
  - Passed as parameter to CBindStatusCallback<>::Download()



334

CBindStatusCallback<>

QA-IQ

- Implements IBindStatusCallback
  - But only GetBindInfo(), OnStartBinding() and OnStopBinding()
- Does not implement IBindStatusCallback::OnProgress()
- To get progress notifications
  - Derive a new class from CBindStatusCallback<>

335

Summary

QA-IQ

- Monikers identify objects and resources
- Binding resolves a moniker into an object
- Binding can be synchronous
  - Or asynchronous
- URL Monikers encapsulate web resources
  - And support asynchronous download
- ATL provides support for asynchronous download

336