Oracle XML DB

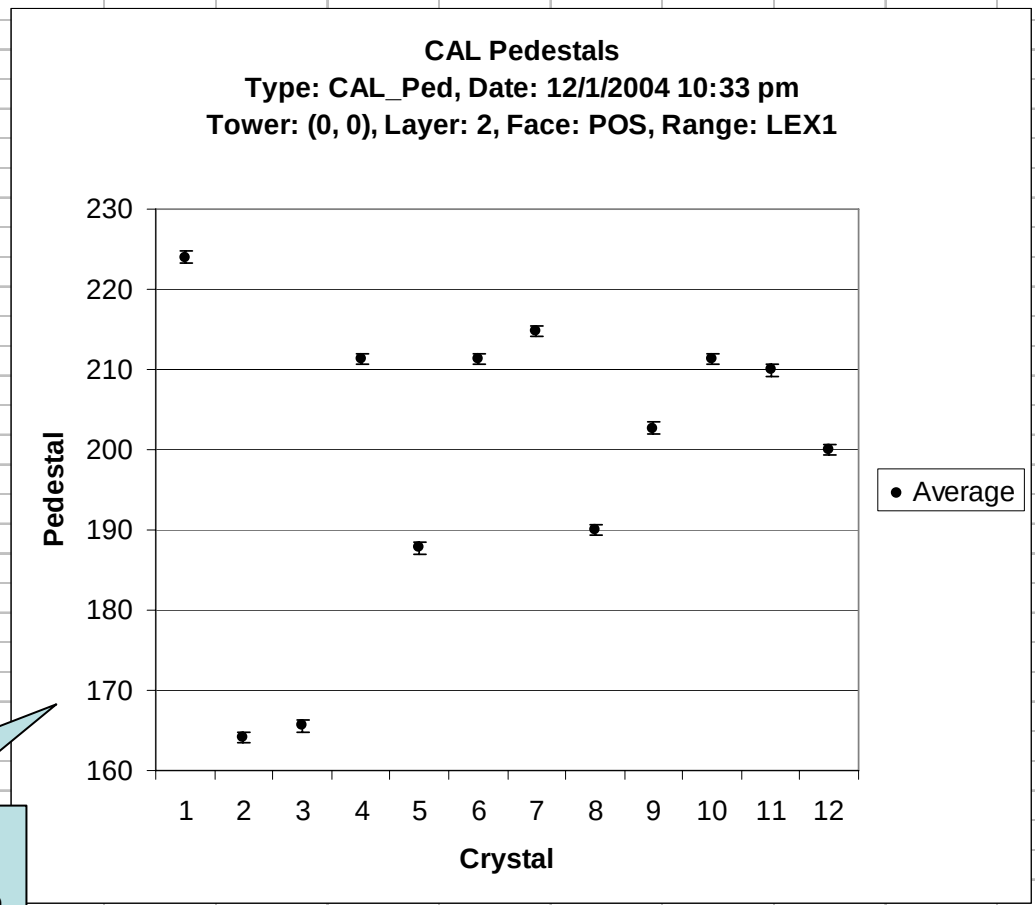
Matthew D. Langston
Stanford Linear Accelerator Center
GLAST SAS Developers Workshop
April 29, 2005

Outline of Talk

- Why Combine XML and Databases?
- XML for pedestrians
  - GLAST Calibration files
  - DTD and Schema
    - database experts call this CREATE TABLE
  - Joanne's DTD  Matt’s Schema
- Oracle XML Database features

Database + Excel

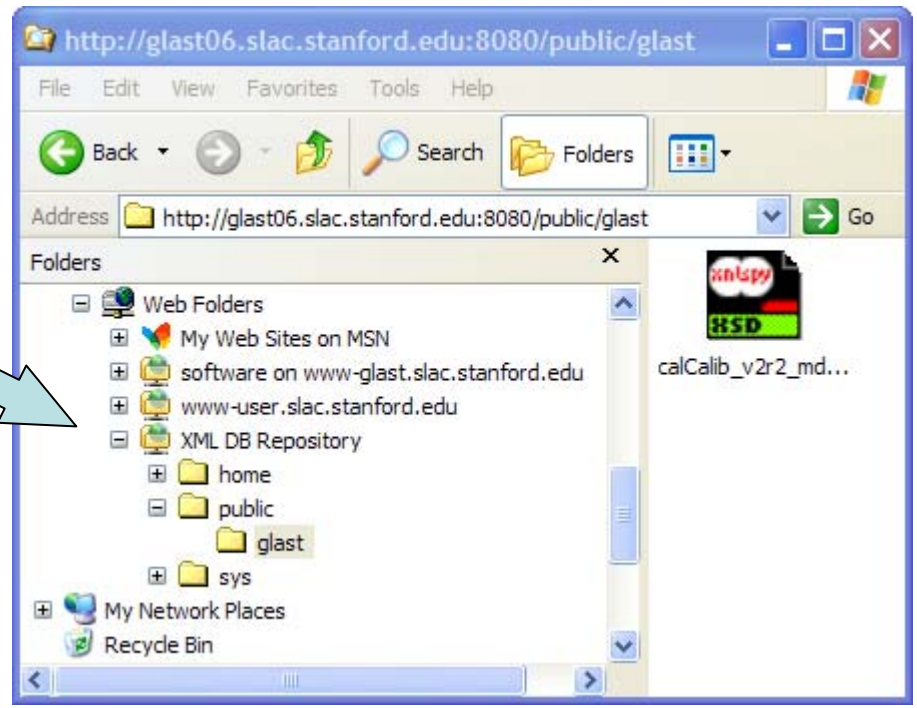
Type	CAL_Ped	
Date	12/1/2004 22:33	
Tower Row	0	
Tower Column	0	
Layer	2	
Face	POS	
Range	LEX1	
	CAL Pedestal	
Crystal	Average	Sigma
1	224.009	0.699958
2	164.114	0.679342
3	165.578	0.693983
4	211.301	0.681233
5	187.718	0.676499
6	211.242	0.688103
7	214.798	0.674496
8	189.965	0.693877
9	202.707	0.678307
10	211.263	0.697349
11	209.934	0.698314
12	199.932	0.691547



Putting XML into Oracle makes it is easy for data analysis

Why Combine XML with a DB?

- Databases Pros
  - Better at storing and searching data
  - Data integrity
  - Central support
- XML Pros
  - Better at representing data
  - All you need is emacs
  - More natural to think in terms of
    - Windows Explorer file system view
    - Linux shell filesystem view



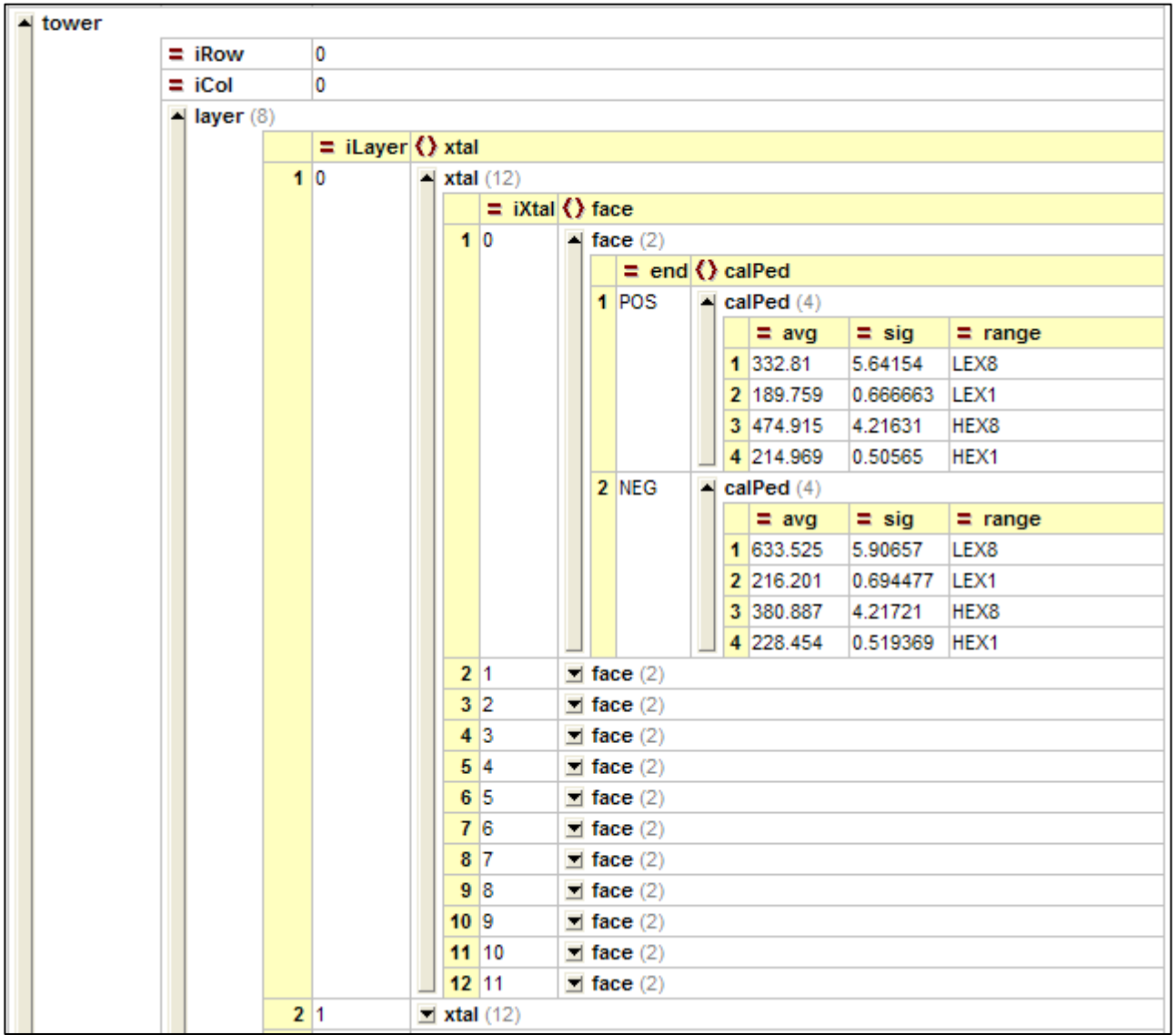
Current Calibration File Strategy

- Use MySQL for “metadata” about XML files in a UNIX filesystem.
  - Metadata is easily queried for.
  - Calibration constants more difficult to get at.
- Issues with current strategy
  - How to get these to the end user’s desktop?
    - Client side software
      - Windows
      - Linux
      - perl?, rsync?
  - How to analyze data?
    - Currently difficult
    - Ad hoc solutions use multiple redundant files.
  - How to trend, or look at time histories, of calibration data?
- Better to store all information in database and generate XML on the fly?

Typical GLAST Calibration File

```
<?xml version="1.0"?>
<calCalib xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://glast06.slac.stanford.edu:8080/public/glast/calCalib_v2r2
_mdl.xsd">
    <generic instrument="LAT" timestamp="2004-12-01T22:33:00" calibType="CAL_Ped"
fmtVersion="v2r2"/>
    <dimension nRow="1" nCol="1" nLayer="8" nXtal="12" nFace="2" nRange="4"/>
    <tower iRow="0" iCol="0">
        <layer iLayer="0">
            <xtal iXtal="0">
                <face end="POS">
                    <calPed avg="332.81" sig="5.64154" range="LEX8"/>
                    <calPed avg="189.759" sig="0.666663" range="LEX1"/>
                    <calPed avg="474.915" sig="4.21631" range="HEX8"/>
                    <calPed avg="214.969" sig="0.50565" range="HEX1"/>
                </face>
                <face end="NEG">
                    <calPed avg="633.525" sig="5.90657" range="LEX8"/>
                    <calPed avg="216.201" sig="0.694477" range="LEX1"/>
                    <calPed avg="380.887" sig="4.21721" range="HEX8"/>
                    <calPed avg="228.454" sig="0.519369" range="HEX1"/>
                </face>
            </xtal>
            <xtal iXtal="1">
                <face end="POS">
                    <calPed avg="439.818" sig="5.64147" range="LEX8"/>
                    <calPed avg="199.699" sig="0.675" range="LEX1"/>
                    <calPed avg="716.009" sig="4.12846" range="HEX8"/>
                    <calPed avg="207.596" sig="0.538755" range="HEX1"/>
```

Typical GLAST Calibration File



tower

= iRow	0
= iCol	0

layer (8)

	= iLayer	{} xtal
1	0	xtal (12)
2	1	xtal (12)

xtal (12)

	= iXtal	{} face
1	0	face (2)
2	1	face (2)
3	2	face (2)
4	3	face (2)
5	4	face (2)
6	5	face (2)
7	6	face (2)
8	7	face (2)
9	8	face (2)
10	9	face (2)
11	10	face (2)
12	11	face (2)

face (2)

	= end	{} calPed
1	POS	calPed (4)
2	NEG	calPed (4)

calPed (4) — POS

	= avg	= sig	= range
1	332.81	5.64154	LEX8
2	189.759	0.666663	LEX1
3	474.915	4.21631	HEX8
4	214.969	0.50565	HEX1

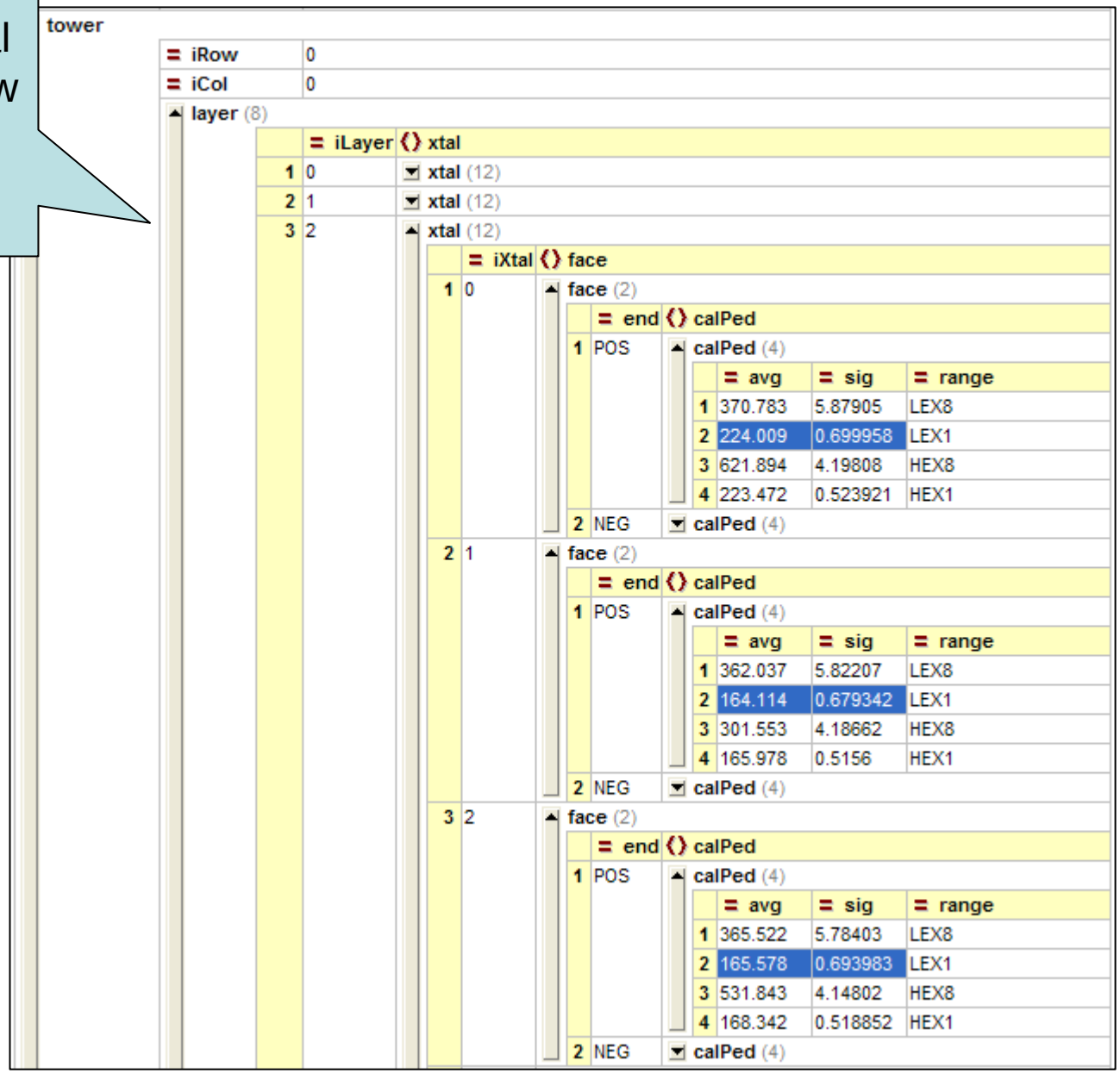
calPed (4) — NEG

	= avg	= sig	= range
1	633.525	5.90657	LEX8
2	216.201	0.694477	LEX1
3	380.887	4.21721	HEX8
4	228.454	0.519369	HEX1

XML naturally describes hierarchical relationships. But, how do you extract this data?

SQL is a “table-based” or “ralational-based” search language

XPath is a “hierarchical” search language.



tower

= iRow	0
= iCol	0

layer (8)

	= iLayer	{} xtal
1	0	xtal (12)
2	1	xtal (12)
3	2	xtal (12)

	= iXtal	{} face
1	0	face (2)
2	1	face (2)
3	2	face (2)

	= end	{} calPed
1	POS	calPed (4)
2	NEG	calPed (4)

	= avg	= sig	= range
1	370.783	5.87905	LEX8
2	224.009	0.699958	LEX1
3	621.894	4.19808	HEX8
4	223.472	0.523921	HEX1

	= avg	= sig	= range
1	362.037	5.82207	LEX8
2	164.114	0.679342	LEX1
3	301.553	4.18662	HEX8
4	165.978	0.5156	HEX1

	= avg	= sig	= range
1	365.522	5.78403	LEX8
2	165.578	0.693983	LEX1
3	531.843	4.14802	HEX8
4	168.342	0.518852	HEX1

Current Calibration File DTD (v2r2)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Matthew Langston (private) -->
<!ELEMENT calCalib (generic, dimension, dac*, xpos?, tower*)>
<!ELEMENT generic (inputSample?)>
<!ATTLIST generic
    instrument (LAT | BFEM | BTEM | EM | EM2 | CU) #REQUIRED
    timestamp NMTOKEN #REQUIRED
    calibType (CAL_LightAtt | CAL_LightAsym | CAL_LightYield | CAL_Ped | CAL_ElecGain |
CAL_IntNonlin | CAL_DiffNonlin | CAL_MuSlope | CAL_Asym | CAL_MevPerDac | CAL_TholdCI |
CAL_TholdMuon) #REQUIRED
    DTDVersion NMTOKEN "v2r2"
    fmtVersion NMTOKEN #REQUIRED
>
<!ELEMENT inputSample (#PCDATA)>
<!ATTLIST inputSample
    startTime NMTOKEN #REQUIRED
    stopTime NMTOKEN #REQUIRED
    triggers NMTOKENS #REQUIRED
    source NMTOKENS #REQUIRED
    mode NMTOKEN #REQUIRED
>
```

My Stab at a Schema (required for some Oracle features)

```
<xs:element name="generic">
  <xs:complexType>
    <xs:attribute name="instrument" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="LAT"/>
          <xs:enumeration value="BFEM"/>
          <xs:enumeration value="BTEM"/>
          <xs:enumeration value="EM"/>
          <xs:enumeration value="EM2"/>
          <xs:enumeration value="CU"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="timestamp" type="xs:dateTime" use="required"/>
    <xs:attribute name="calibType" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="CAL_LightAtt"/>
          <xs:enumeration value="CAL_LightAsym"/>
          <xs:enumeration value="CAL_LightYield"/>
          <xs:enumeration value="CAL_Ped"/>
          <xs:enumeration value="CAL_ElecGain"/>
          <xs:enumeration value="CAL_IntNonlin"/>
          <xs:enumeration value="CAL_DiffNonlin"/>
          <xs:enumeration value="CAL_MuSlope"/>
          <xs:enumeration value="CAL_Asym"/>
          <xs:enumeration value="CAL_MevPerDac"/>
          <xs:enumeration value="CAL_TholdCI"/>
          <xs:enumeration value="CAL_TholdMuon"/>
```

XML Database Benefits

- Searching using a combination of SQL and XPath.
- Integration
  - Easier to exchange data
    - You define the data types, structure, etc.
    - No DB Drivers, etc.
- Automation
  - Insert data into Oracle with “drag-and-drop” using WebDAV, FTP, HTTP, etc.

Searching using Combination of SQL and XPath

Would you rather use this?

```
select calPed.avg, calPed.sig, from calMetaData inner join (calPed inner
join (face inner join (xtal inner join (layer inner join (tower on
layer.tower_fk = tower.tower_pk) on xtal.layer_fk = layer.layer_pk) on
face.xtal_fk = xtal.xtal_pk) on calPed.face_fk = face.face_pk) on
calPed.metaData_fk = metaData.metaData_pk where metaData.calibType =
'CAL_Ped' and metaData.timeStamp = to_date('2004/12/01', 'yyyy/mm/dd')
and tower.iRow = 0 and tower.iCol = 0 and layer.iLayer = 2 and face.end
= 'POS' and calPed.range = 'LEX1'
```

Or this

```
select calPed.avg, calPed.sig, from calMetaData
where existsnode(xml,
'//generic[@timestamp = xs:date("2005-04-23")]' and
'//tower[@iRow = 0 and @iCol = 0]/layer[@iLayer = 2]/xtal/face[@end = "POS"]/calPed[@range = "LEX1"]'
```

- XPath is a rich search language for data organized hierarchically
  - Relatively simple to learn
  - Hundreds of build in functions
  - User defined functions
  - Extensible through scripting

Conclusions

- Oracle 10g is an XML database
  - has had since Oracle 8i
  - Probably the best in the world
- SLAC needs to enable XML DB Repository
  - Allows inserting data using drag-and-drop using WebDAV, FTP, HTTP, etc.