



December 3-6, 2007, Santa Clara Marriott, Santa Clara, CA

sfc small footprint cim broker

Sven Schuetz
IBM

Presented by: Ed Boden, IBM



Agenda

- sfcB
 - Overview, features
 - Some implementation details
- sfcC
 - Overview
 - Architecture
 - Connection between sfcB and sfcC
- New features



sfc

Small footprint CIM broker



Overview sfcb

- sfcb is a CIM server for embedded environments
- Meets typical embedded requirements
 - Low resource consumption
 - Robustness (unattended operation)
 - Configurability (features/footprint)
- Open Source package available under the Eclipse Public License (EPL) from the SBLIM project
- Supports CMPI providers



Features

- Performance
 - Single relocatable object format used on disk, in memory and on the wire reduces marshalling overhead
 - Preallocated socket pairs for low-latency connections
 - No central dispatcher for requests and data



Features

- Robustness
 - Central Process doesn't load foreign code (like providers, adapters)
 - Configurable Provider to Process Mapping allows maximum isolation
 - Protocol Adapters (Codecs) can run in separate processes

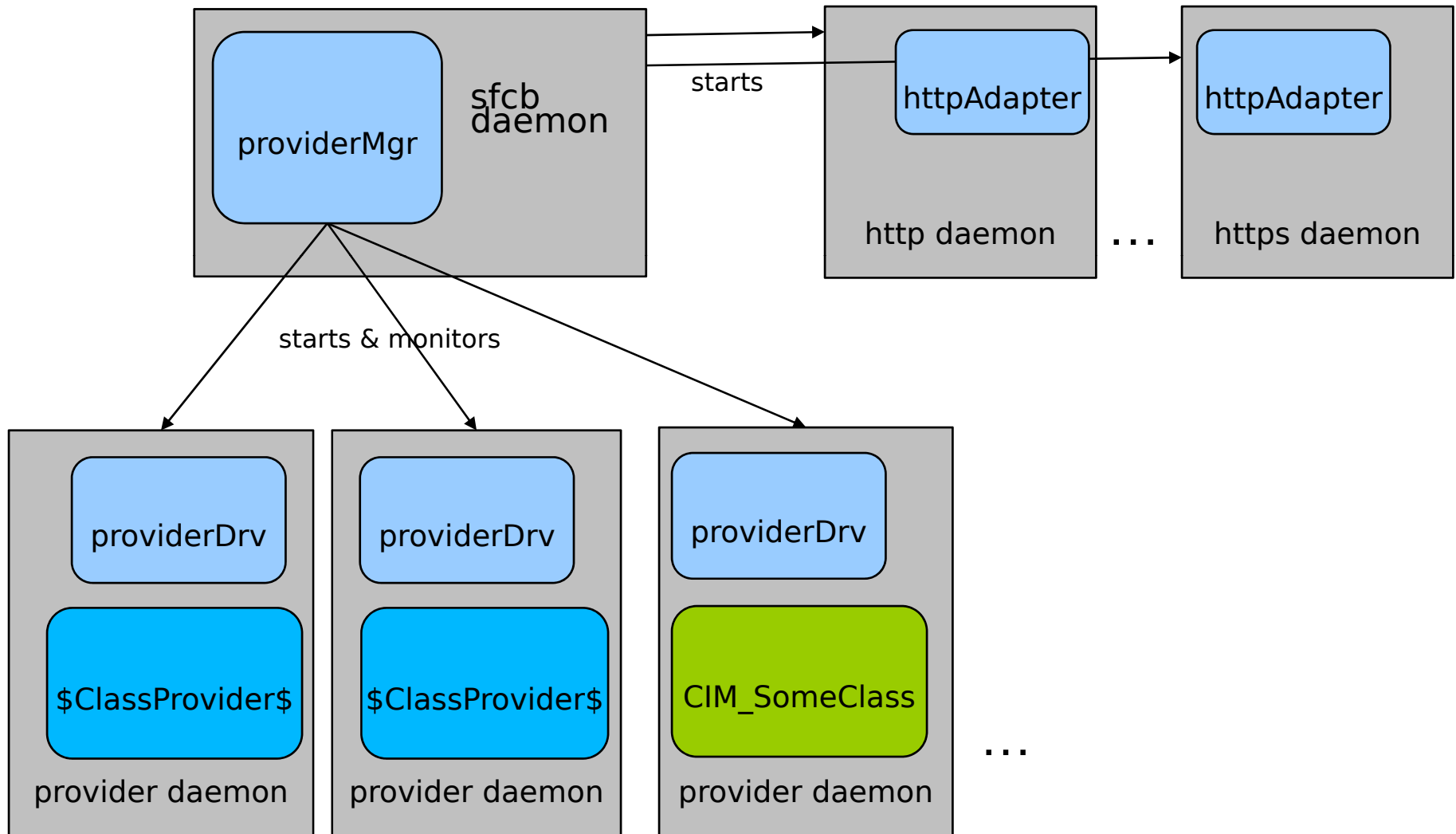
- Configurability
 - Build Time Options (Feature Capability)
 - Run Time Options (Feature Usage)
 - Extension Providers
 - Adapters/Codecs



Implementation

- Core process acts as Provider Manager
 - Maintains provider registration data
 - Resolves and starts providers on request from adapters and up-calls
 - Everything else is adapter or provider
- Adapter/Codec processes
 - Receive and decode client requests
 - Send decoded requests to providers
 - Receive provider responses and encode into client format
- Provider processes
 - Load CMPI provider libraries
 - Receive requests by adapters/providers and send back responses to caller

Implementation - Components



Adapters

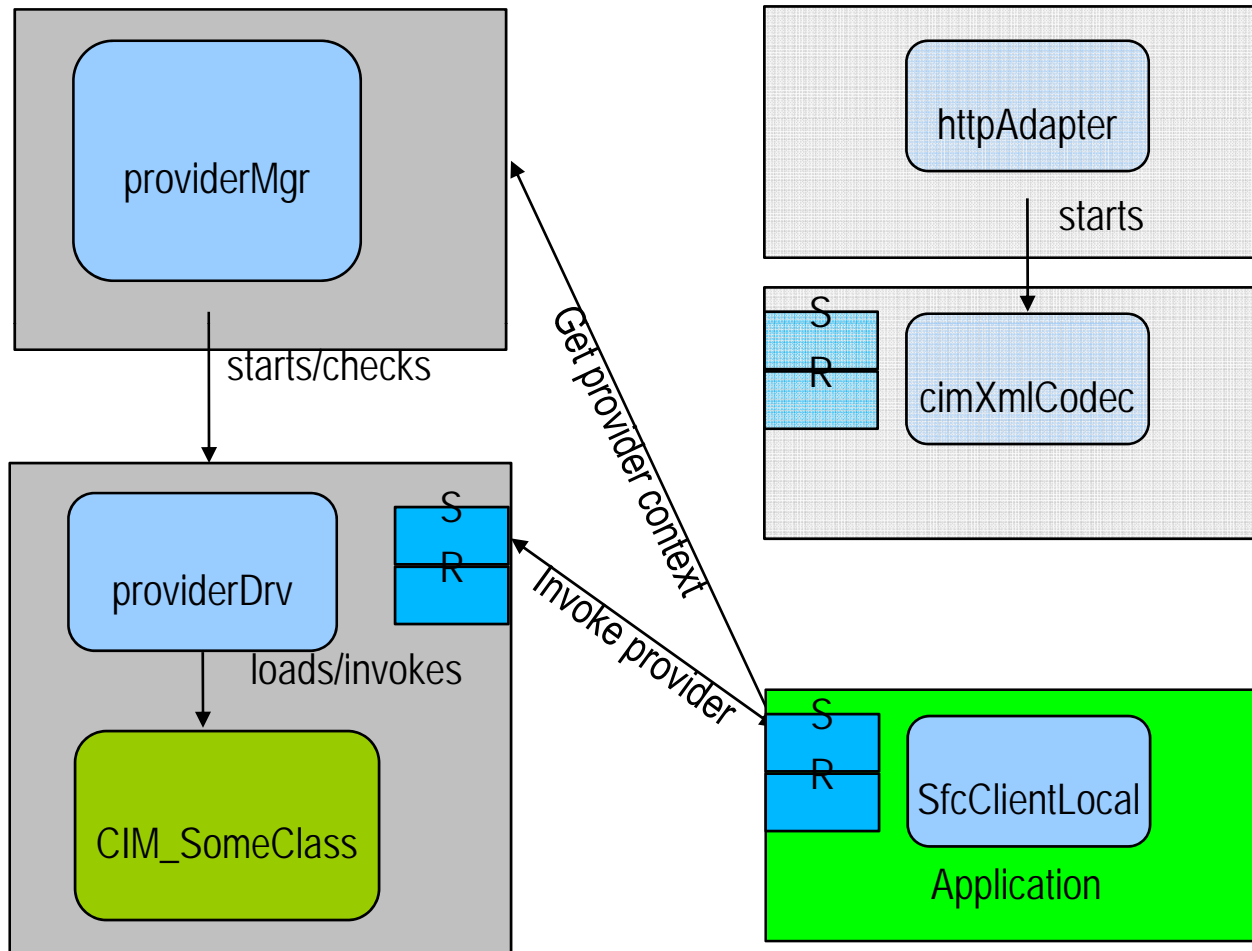
- Adapters are required for communication between clients and sfc
- If a protocol data translation is required we speak also of codecs
- Sfc comes with an http/https adapter (with an CIMXML codec)
 - and an *experimental* JDBC adapter
- Local client communication will also use the adapter communication mechanism



Local Client support

- Based on internal communication protocol
- Using sfcb object format
- Implemented as backend for sfcc (small footprint CIM client)

Local Client as Adapter



Footprint numbers

	Disk (kb)	RAM (kb)	Shared lib (kb)
Minimum config	588	1176	400
Default w/o Schema	1016	2058	900
Default with Schema	4260	2058	900



sfcc

Small footprint CIM client library

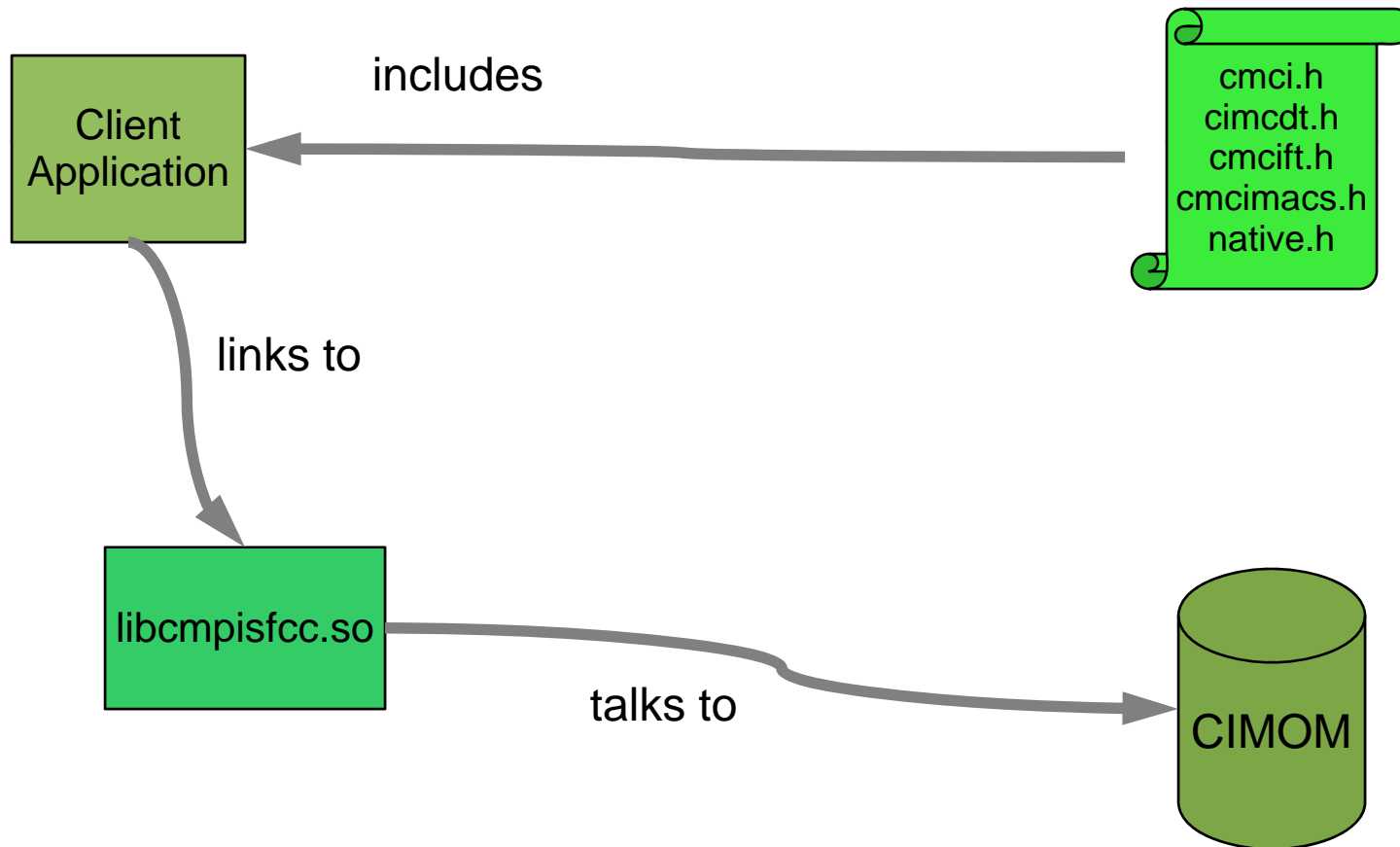


sfcc overview

- C API allowing client applications to interface with CIM implementations
- Small memory and disk footprint - well-suited for embedded environments
- Based on CMPI
- Initial version released 11/2006
- Second version released 02/2007
 - New architecture
 - Modular

sfcc version one

Simple architecture overview





Sample code version one (deprecated)

```
CMClient          * cc;
CMObjectPath     * objectpath;
CMEnumeration    * enumeration;
CMStatus         status;

/* Setup a connection to the CIMOM */
cc = cmciConnect("localhost", "http", "5988", "user", "pw", &status);

/* Test enumInstanceNames() */
objectpath = newCMObjectPath("root/cimv2", "CIM_Process", NULL);

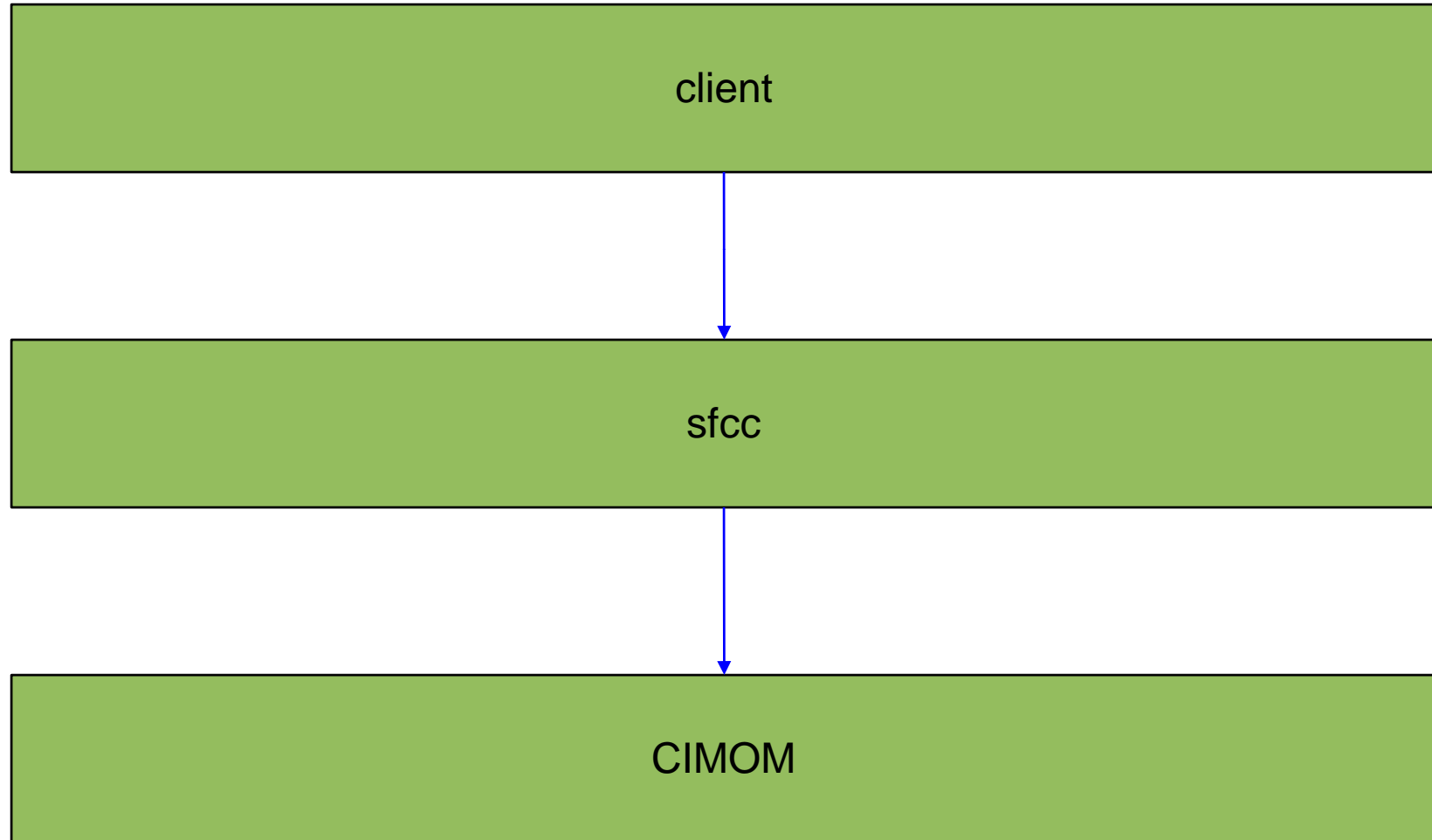
enumeration = cc->ft->enumInstanceNames(cc, objectpath, &status);

/* Print the results */

if (!status.rc) {
    printf("result(s): \n");
    while (enumeration->ft->hasNext(enumeration, NULL)) {
        CMData data = enumeration->ft->getNext(enumeration, NULL);
        showObjectPath(data.value.ref);
    }
}

if (enumeration) CMRelease(enumeration);
if (objectpath) CMRelease(objectpath);
if (cc) CMRelease(cc);
if (status.msg) CMRelease(status.msg);
```

sfcc Layers version one

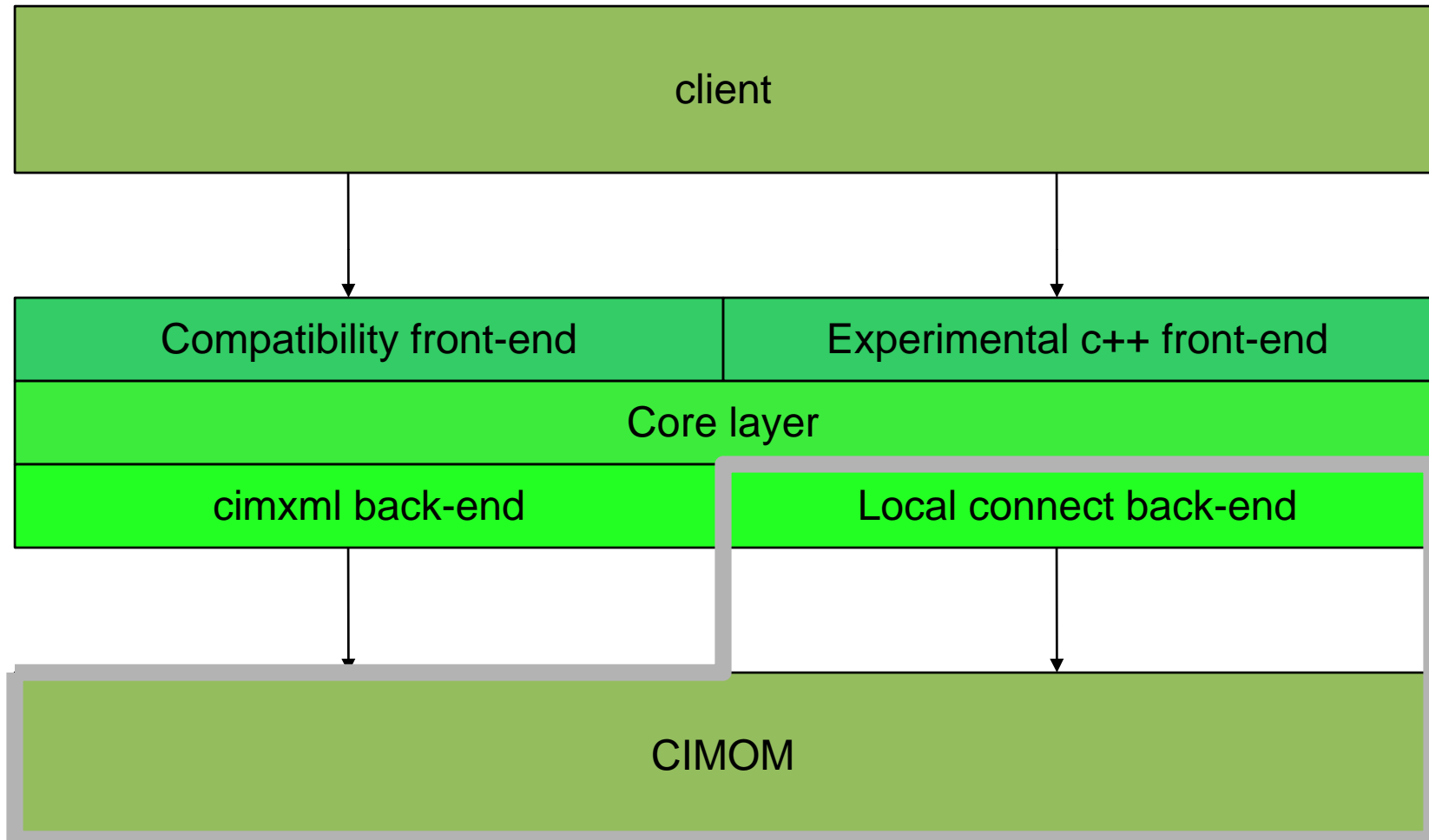




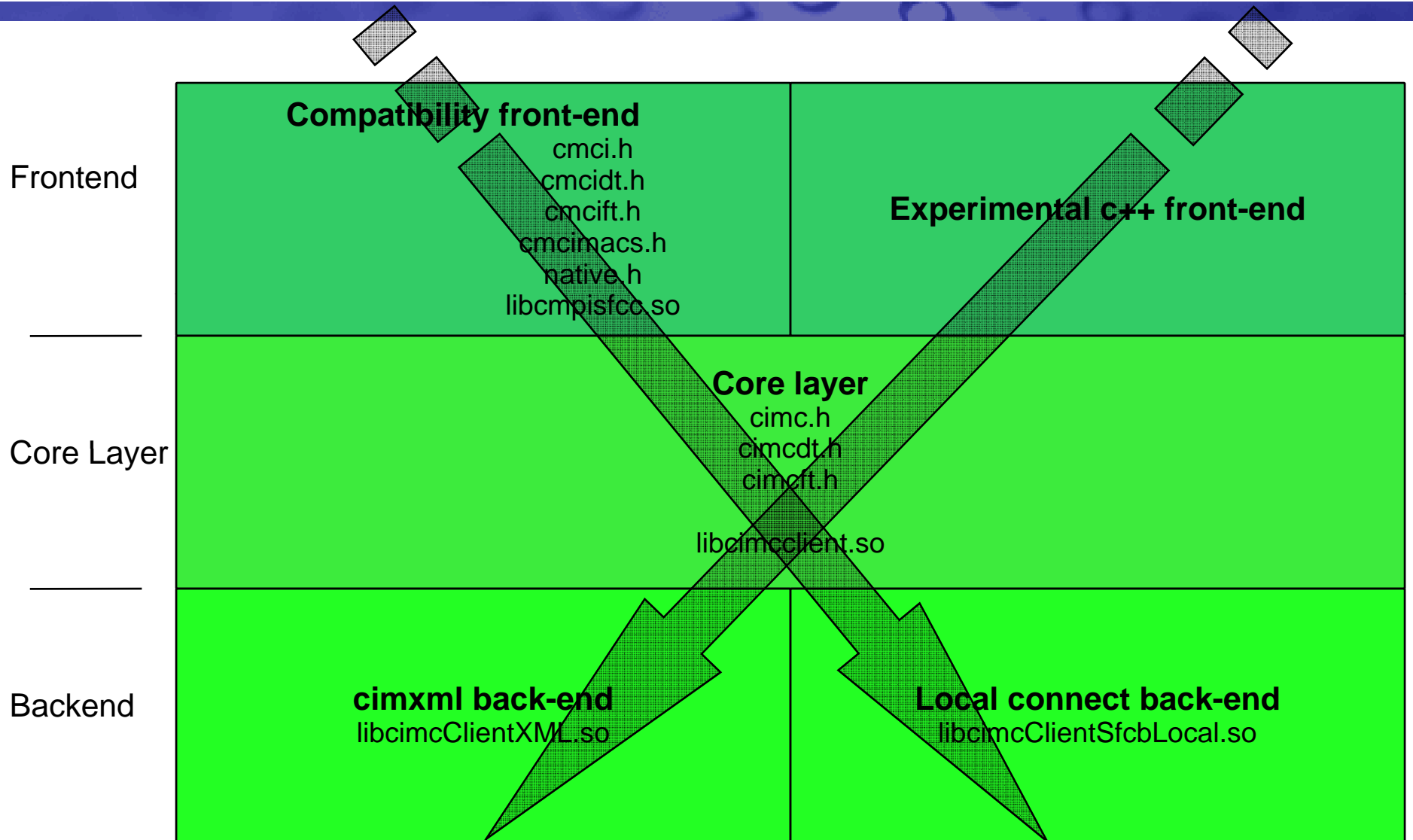
sfcc overview version two

- Layered approach
 - Front-Ends
 - Core layer
 - Back-Ends
- Two back-ends available
 - cimxml
 - Local connect
- Two front-ends
 - Compatibility front-end
 - Experimental c++ front-end

sfcc Layers version two



sfcc layers in depth





Sample code version two, core layer

```
CI MCEnv *ce;
char *msg;
int rc;
CI MCStatus status;
CI MCCli ent *cli ent;
CI MCObj ectPath *op;
CI MCEnumerati on *enm;

ce = NewCI MCEnv("XML", 0, &rc, &msg);

cli ent = ce->ft->connect(ce, "host", "http", "5988", "user", "pw", &status);

op = ce->ft->newObj ectPath(ce, "root/ci mv2", "CI M_ComputerSystem", &status);
enm = cli ent->ft->enumI nstanceNames(cli ent, op, &status);

if (!status.rc) {
    printf("resul t(s): \n");
    while (enm->ft->hasNext(enm, NULL)) {
        CI MCData data = enm->ft->getNext(enm, NULL);
        showObj ectPath(data.val ue.ref);
    }
}
ce->ft->rel ease(ce);
cli ent->ft->rel ease(cli ent);
op->ft->rel ease(op);
enm->ft->rel ease(enm);
```

back-end library

A blue arrow points from the text 'back-end library' to the 'NewCI MCEnv' function call in the code.

Connection sfcb/sfcc Advantages



- sfcb & sfcc – perfect duo
- Work well in resource constraint environments
- Tested with each other
- Local connect only possible between sfcc & sfcb
- Same team, synchronized releases



Connection sfcb/sfcc Developer benefits

- Cim clients and servers have overlapping tasks
 - Parsing
 - Internal object format
 - ...
- Development of sfcc was fast, some components (e.g. the parser) have been copied
- Potential to save work (was done) and to save code (was not done)
 - Shared libraries possible

New Features



New Features

sfcb 1.3, sfcc 2.1

- CMPI 2.0 (sfcb)
- Execution time parameters to providers (sfcb)
- Large Volume Support
 - Cimxml (sfcc)
 - local connect (sfcb, sfcc)
- Indication support
 - Cimxml (sfcc)
 - local connect (sfcb / sfcc)



Execution time parameters

- Option in config file
- Passed via the CMPI Context

Provider registration:

```
[$ClassProvider$]  
  provider: ClassProvider  
  location: sfcClassProviderGz  
  type: class method  
  unload: never  
  namespace: *  
  parameters: nocache
```

Provider:

```
ctx->ft->getEntry(ctx, "sfcbProviderParameters", &status);
```



Large Volume Support

(sfcc)

- Client can begin parsing and evaluating the response even when the server is still busy
- For huge requests
 - Saves time
 - Saves resources (mainly memory)
- Implemented via an iterator, getNext in the enumeration
- Only for enumerative requests



Large Volume Support

(sfcc)

- CIMXML
 - HTTP Chunking is used
 - sfcc uses curl for http communication
 - Curl supports chunking, has a callback mechanism
 - Curl callbacks invoke XML parser and start processing
 - Client can start enumerating the responses as soon as the first chunk is received
- Local Connect
 - No network protocol, easier implementation



Indication Support

(sfcc)

- Realized via CMPI Style “object” with function pointers – CIMCIndicationListener
- CIMXML
 - sfcc acts as an HTTP Server
 - Client registers callback function which is called upon indication reception
 - Redesign of parser was necessary (switch from Bison to own parser), as Bison needs the complete message
- Local connect
 - Client listens on a local socket
 - Callback function



Future considerations

- OpenWSMAN support via local connect
- Complete Threadification (In prep for Embedded, Windows)
- Support for new ‘pulled enumerations’
 - sfcb, sfcc
 - DMTF WIP CR 000386
- Footprint reduction
- Increased test automation
- For latest info:
<http://sblim.wiki.sourceforge.net/SfcbRoadmap>



References

- Links
 - SBLIM Open Source Project Homepage
<http://sblim.sourceforge.net>
 - SourceForge URL (Downloads, Bug Reports)
<http://sourceforge.net/projects/sblim>
- Trademarks

IBM is a trademark of IBM Corporation in the USA and/or other countries

Other company, product, or service names may be trademarks or service marks of others.