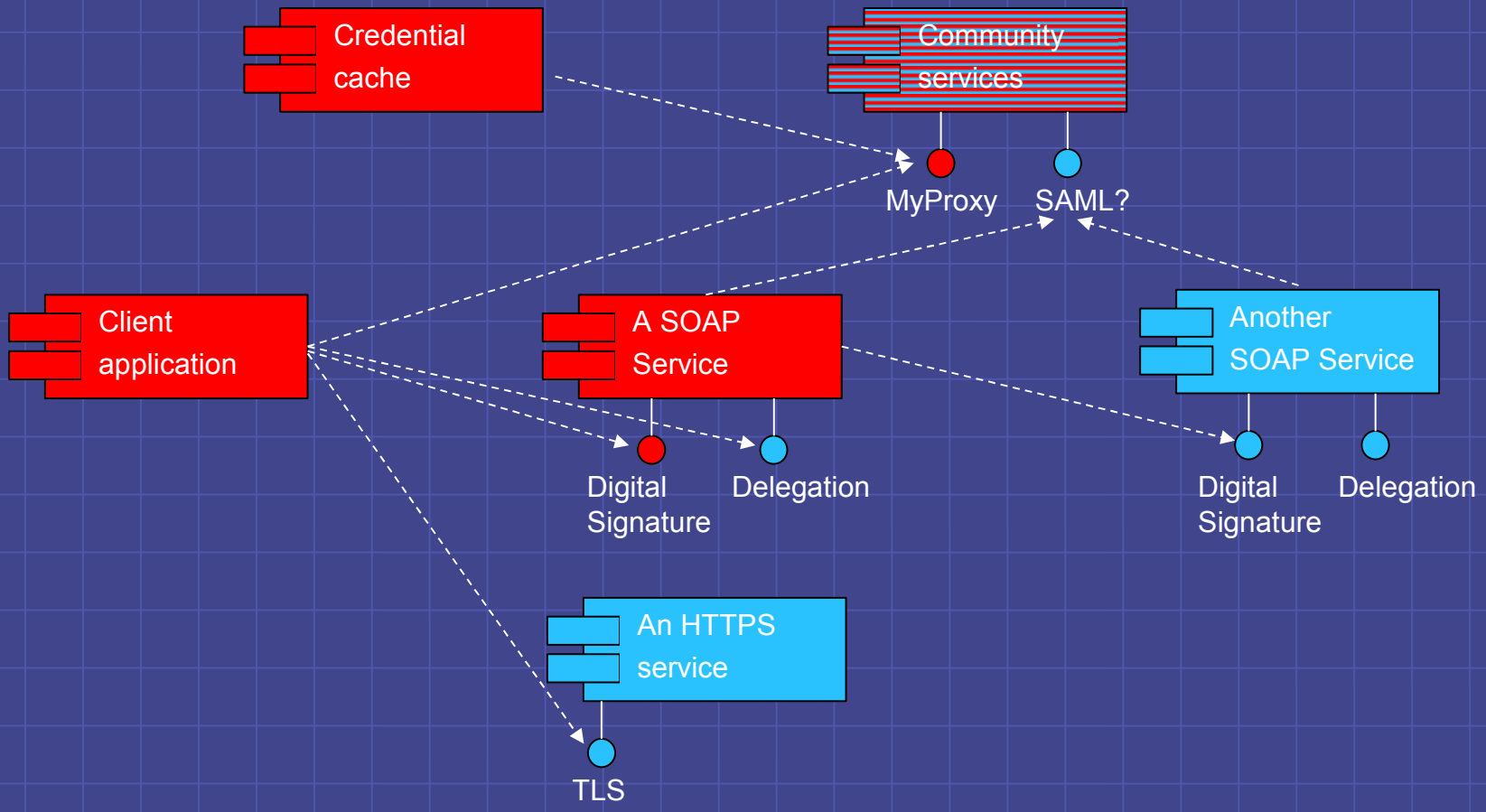
IVOA/AstroGrid SSO system and Grid standards

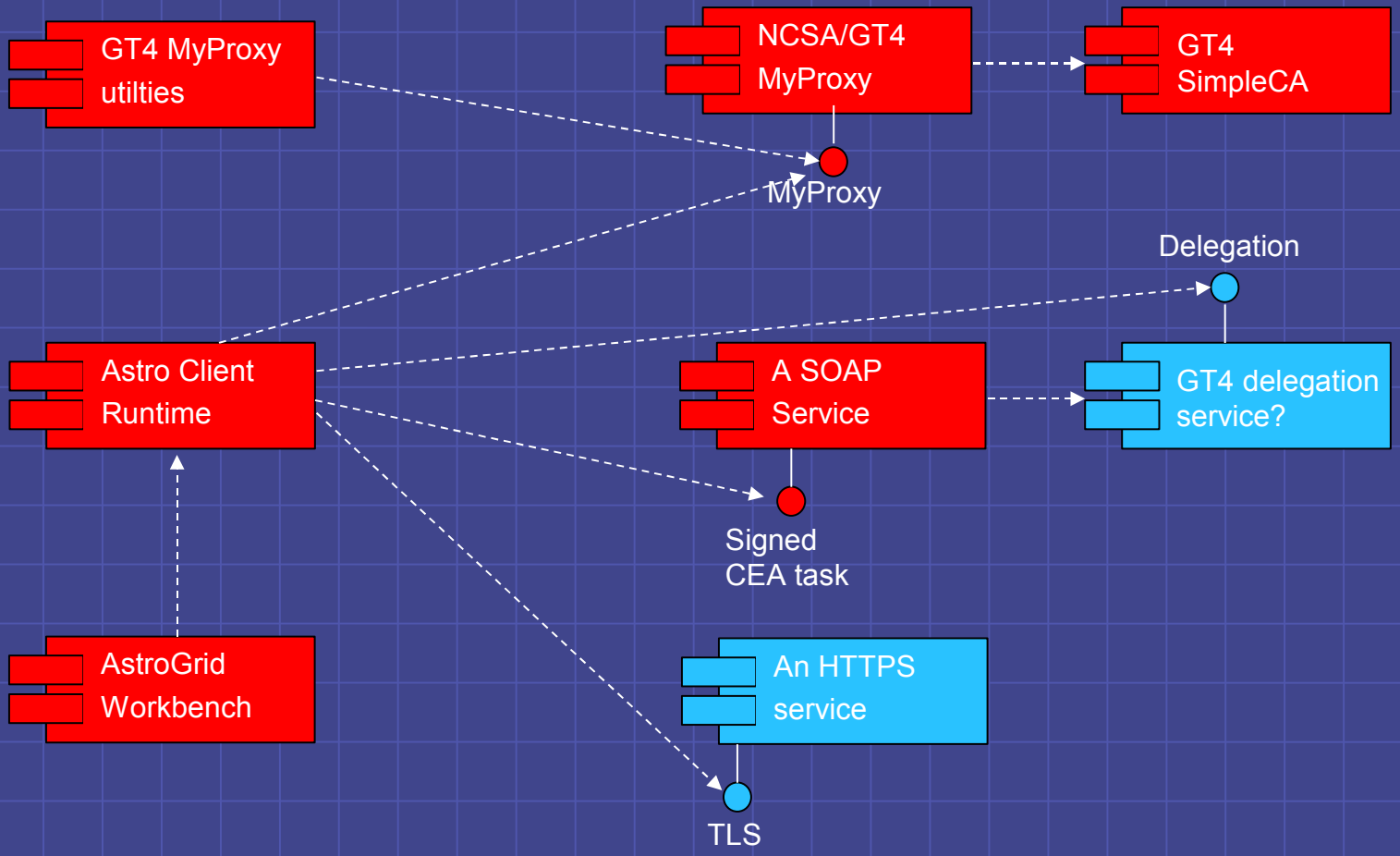
Guy Rixon and Keith Noddle

Presentation to Astro-RG at GGF17

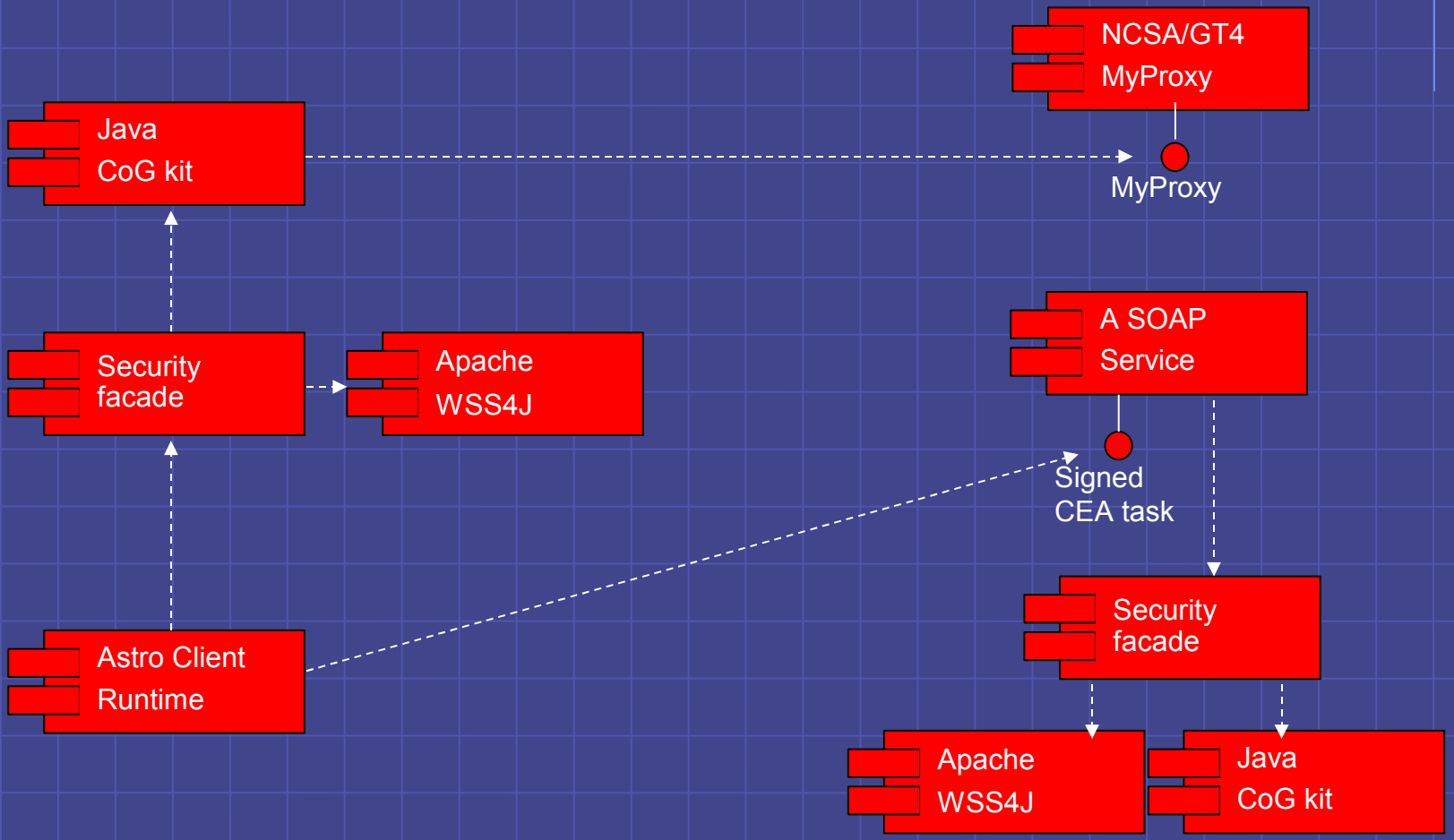
The IVOA SSO scheme



AstroGrid implementation (1)



AstroGrid implementation (2)

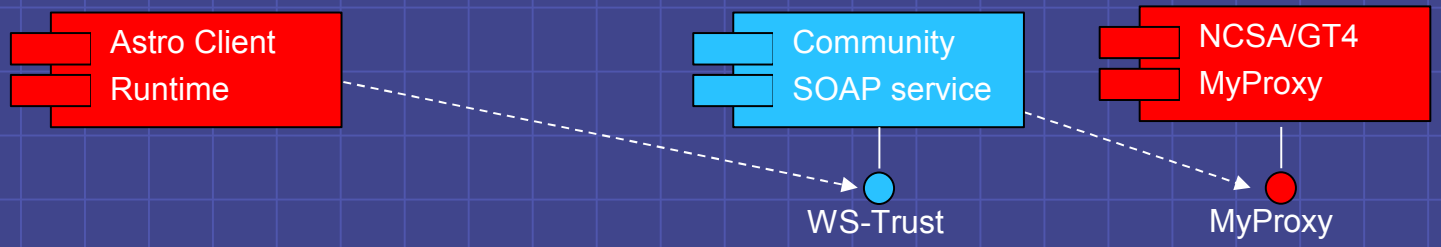


AstroGrid/IVO overlap with Grid

- Principles:
  - X.509 certificates
  - RFC3820 proxies
  - PKI including national science CAs
  - MyProxy
- Reused software:
  - MyProxy, SimpleCA from GT4
  - RFC3820, trust-anchor support from Java CoG kit
- Possible further reuse:
  - Delegation service from GT4
  - TLS support from Java CoG kit
  - Attributes: PERMIS? VOMS?

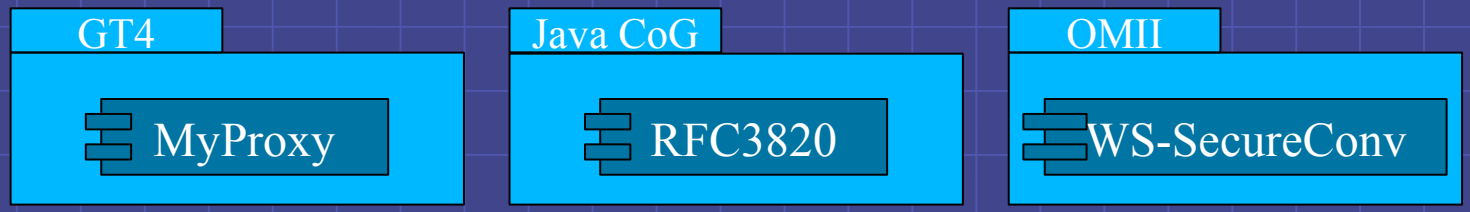
Departure from Grid convention

- Not built entirely within Grid toolkit
- No authentication of services (except TLS)
- TLS, but not GSI
- Community-based CA
- May augment MyProxy with WS-Trust

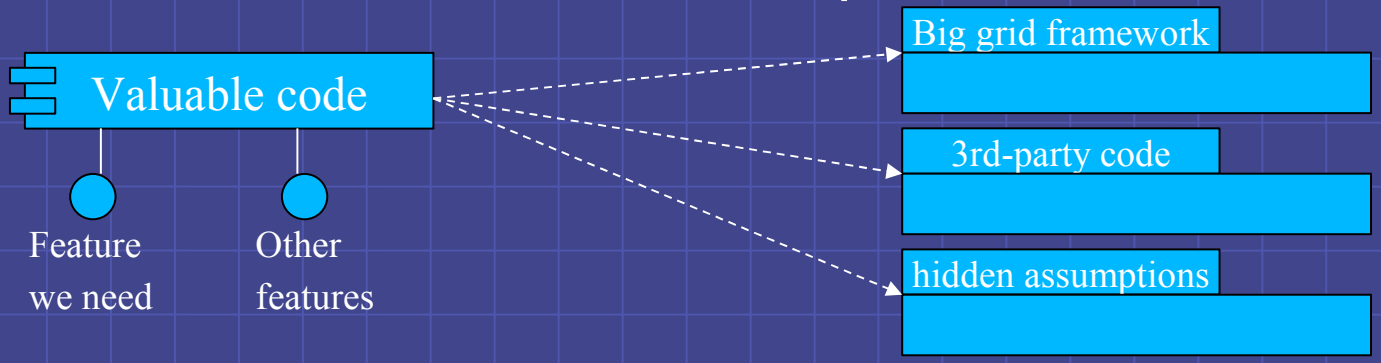


Standard support

- Little support for Grid standards in mainstream tools
- Plenty support in Grid toolkits, e.g.…

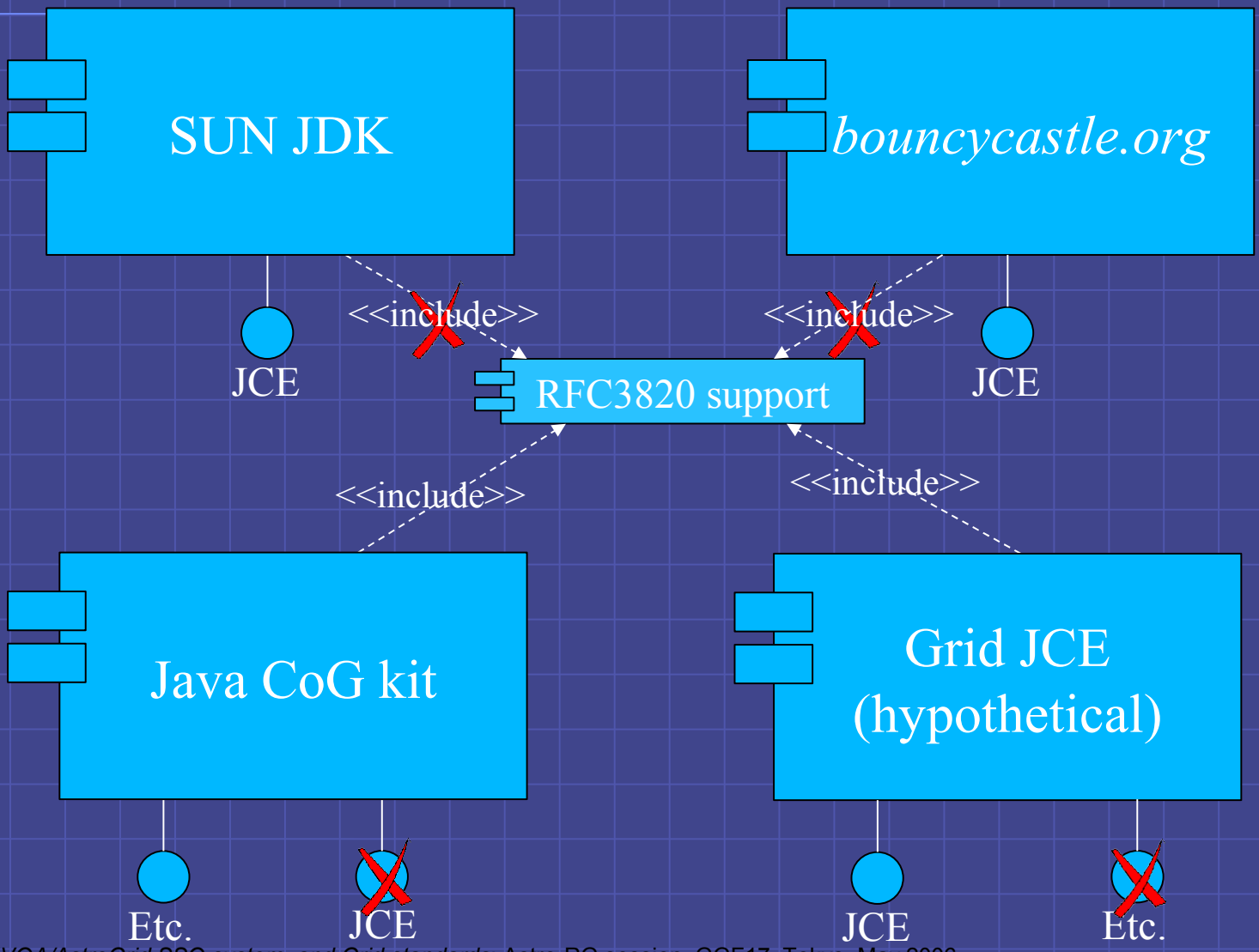


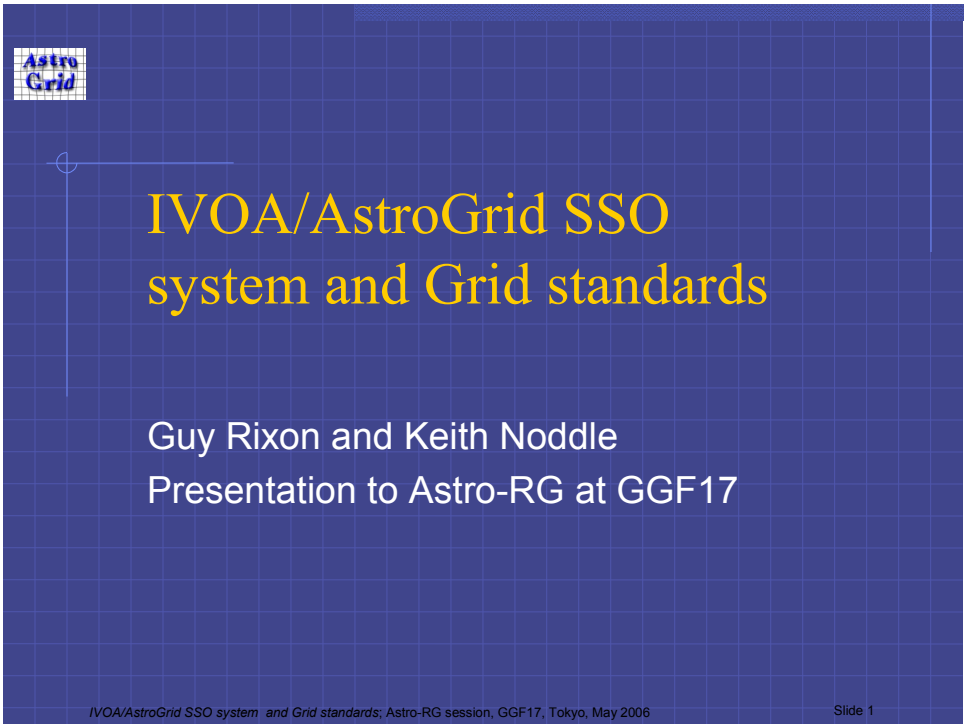
- …but it's hard to extract the parts.



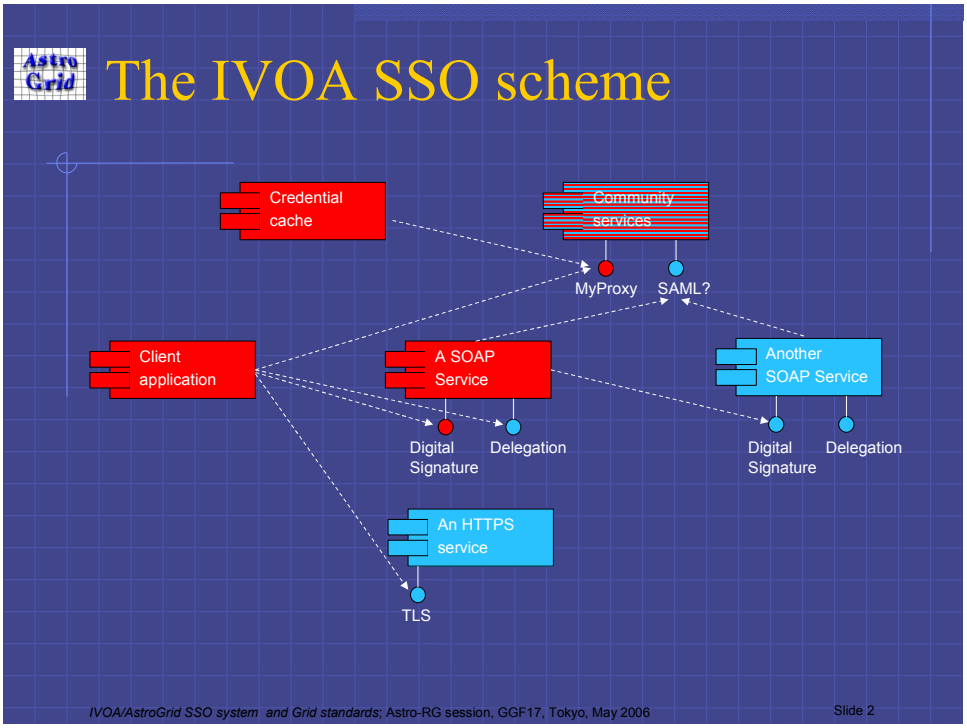
- Is this slowing adoption of the standards?
- Could we have some more-separable, loosely-coupled solutions, please?

Proxy-certificate example





This talk tells how a grid-compatible single-sign-on (SSO) system was concocted in a grid-neutral forum (IVOA) and how a “country cousin” (AstroGrid) of the core GGF movement managed to implement (some of) it, eventually, using code recycled from the grid toolkits.

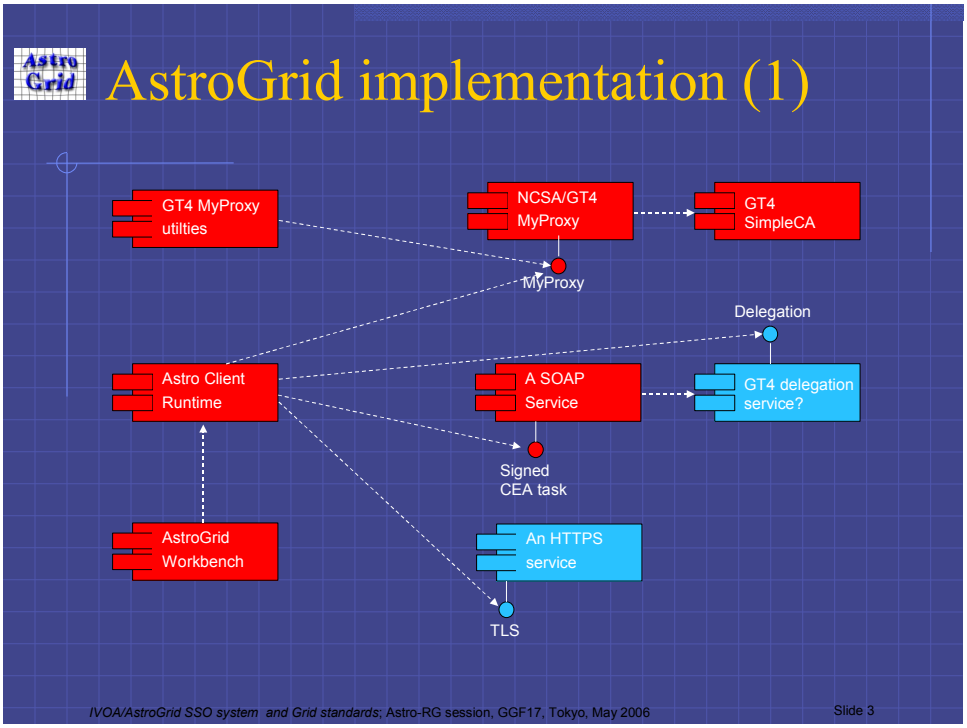


This slide shows the architecture of the IVOA single-sign-on (SSO) system adopted (and partly developed) by AstroGrid.

A client application – desktop GUI or web portal - signs on to the IVO via a MyProxy service run by its user's home institution. The client receives a proxy certificate conforming to RFC3820 which it holds in memory for the duration of the interactive session. The proxy certificate allows the client to sign digitally messages to web services (following the WS-Security standard) or to use TLS to secure a connection to an HTTPS service. The client may also delegate proxy credentials to a web service such that that service may act as the user's agent in calling other services. Attributes of the user's position in the astronomical community – typically membership of user group with specific access rights – can be passed to services from attribute servers run by the user's home community. The mechanism for this is not yet chosen by IVOA; SAML attribute servers are a possibility.

The proxy credentials may be generated from permanent credentials held by the user and the proxy put into MyProxy; or the permanent credentials may be stored in MyProxy; or the proxy credentials may be generated inside MyProxy using a certificate authority (CA) managed by the community. The method used for any given IVO user depends on the user's preferences and the rules of usage of the relevant CAs. E.g. some CAs do not allow permanent credentials to be put in a third-party MyProxy service; some users may prefer to manage their permanent credentials on their own system; some users may prefer to register with their local community and not with a national-level CA. MyProxy makes all these approaches equivalent from the point of view of the client application.

This scheme is not new; it is basically unchanged since the outline agreements made in Kyoto in Spring 2005; the diagram is adapted from an AstroGrid paper presented at ADASS2005. What is new is (a) that the detailed protocols are now being recorded and (b) that some of the elements are now implemented in the IVO. The parts that AstroGrid has implemented (as prototypes) are coloured red and the unimplemented parts are blue (and this convention is used in subsequent slides). The prototype implementation has taken a long time to appear! In part, this is because of the patchy support for the Grid standards involved; and we shall return to this at the end of the talk.



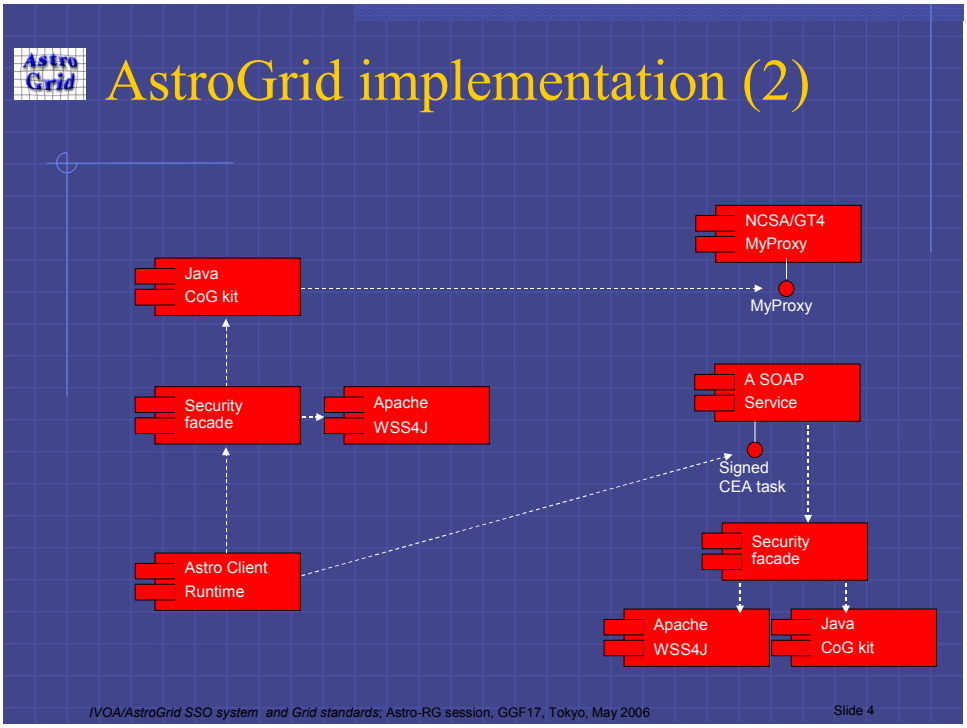
The IVOA schema is fairly general. This slide shows how it has been fitted into the AstroGrid architecture.

The MyProxy server, the local CA software and the utilities for managing certificates on the desktop come from GT4. These components can be used as supplied; they suit the IVO's use-cases very well.

The client-side code for signing SOAP messages, for TLS connections and for MyProxy has been added to the Astro Client Runtime (ACR) component. This component underlies AstroGrid's Workbench UI and is promoted as an abstraction layer for other desktop UI programmes. An ACR instance operates as a local service on its host desktop and can be shared between programmes. Thus, a proxy certificate obtained via one UI is held in the ACR and may be used by other UIs: SSO is achieved.

Servers in the Common Execution Architecture (CEA) are enhanced to be able to check digital signatures on messages.

We still need an implementation of the delegation interface. It is quite likely that we shall take the delegation service from GT4.



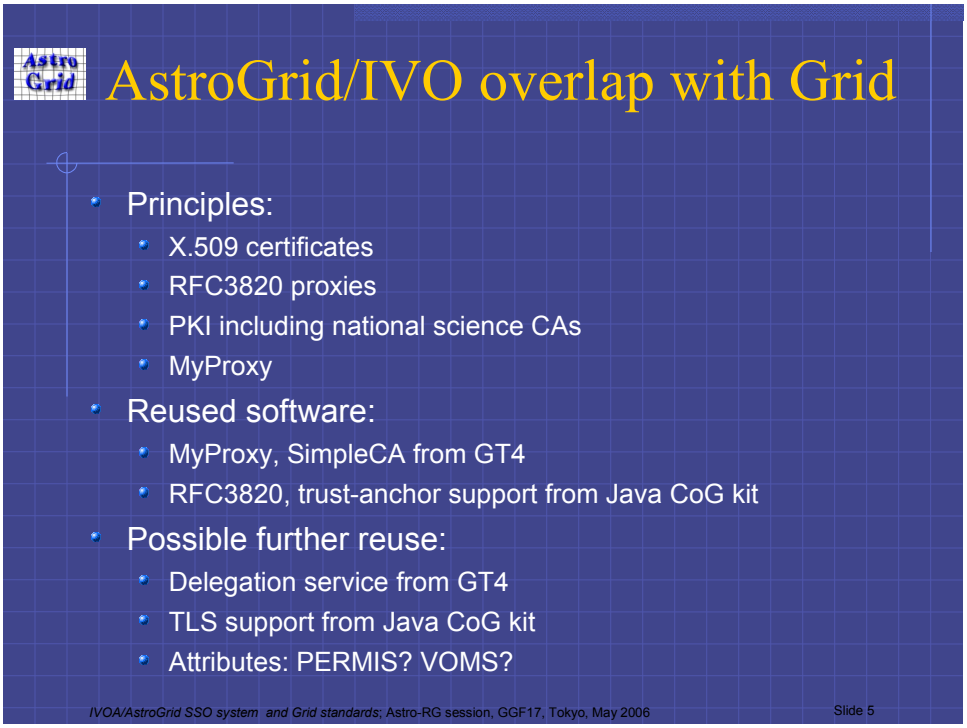
This slide shows some details of the MyProxy and digital-signature implementations.

We get the MyProxy client from the Java CoG kit; as noted above, the server comes from GT4.

Most of the digital signature code is in a highly-customized version of Apache WSS4J, the security extension for Apache Axis. This is difficult code to work with, and not well-matched to our architecture and use-cases, but OSS implementations of WS-Security are few and far between.

The crucially “griddish” part of the system is the use of proxy certificates according to RFC3820. WSS4J does not support RFC3820, even to the extent of recognizing a proxy certificate and rejecting it with a helpful error-message. Therefore, we replace the part of WSS4J that checks certificate chains with that from the Java CoG kit.

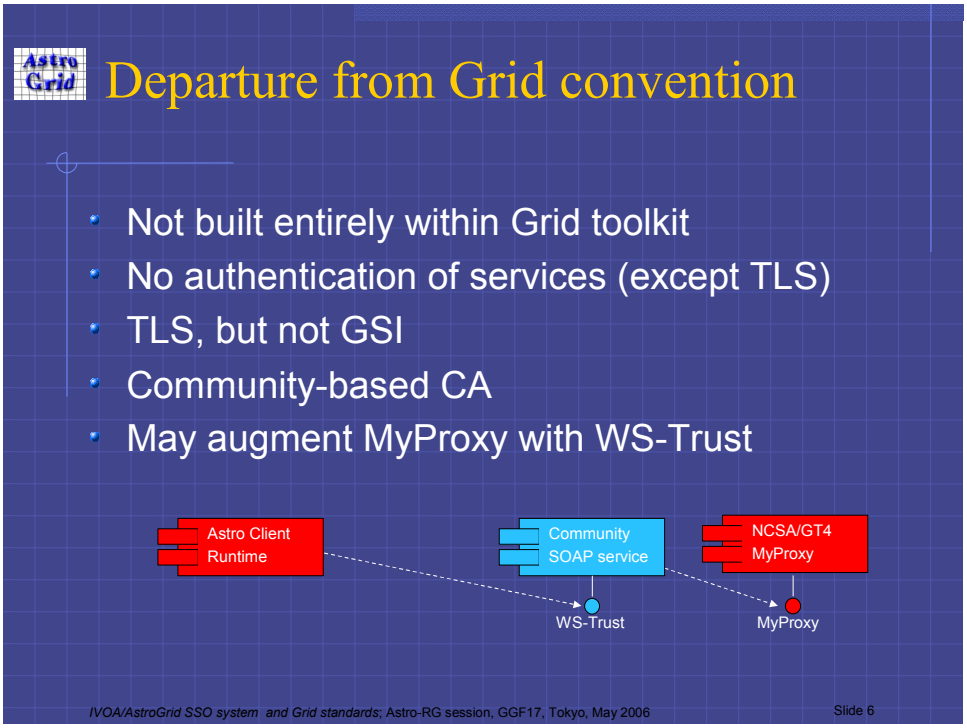
To hide the details of the composite implementation, we provide a security-façade library.



Most of the security concepts in our system come from Grid practice. We've managed to build the current prototypes using Globus and Globus-related parts. As we extend the implementation to cover delegation, TLS and attributes services, then we hope to use more proven, grid software.

We've tried to be good citizens of the Grid movement and have used Grid-friendly methods wherever possible. We want our grid of astronomy applications (in which the commodities are specific to astronomy) to be a client of the general compute grids for science, and we recognize that the difficult part of this connection is the security. Therefore, we've deliberately chosen methods, structures and software from the grid world *where our use cases allow this.*

However… *(segue to next slide for the flip side of the argument)*



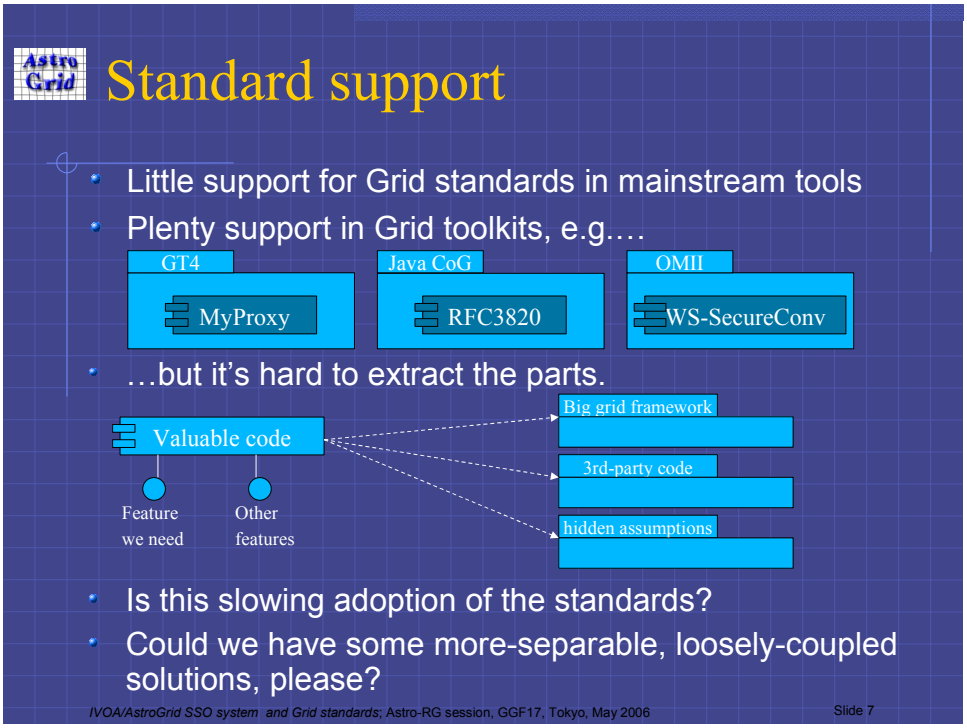
(continue commentary from previous slide) … however, our part of the IVO is *not* purely a compute grid. We are not in a position to build it soley with established grid tool-kits and our climax state is *not* purely an OGSA-compatible grid. Thus, we depart in places from the common conventions.

We do not do *mutual* authentication in requests to servers. Our use-cases do not demand it and, by leaving out authentication of servers, we reduce the need to issue and maintain service credentials.

We intend TLS but not GSI for talking to HTTPS services, in order to ease the implementation and have broader base of reusable code. We might be persuaded to reverse this decision; but IVOA would need persuasion too.

We still intend to support local, community operated CAs, despite warnings from Grid people. We see this as at least a necessary, temporary step until all IVO users are also Grid users.

We like MyProxy but have problems with the ports it requires. The rest of our system is based entirely on HTTP and we only require participating sites to open ports 80, 8080 and 443; MyProxy complicates this. We might, in the medium term, prefer to replace (or augment) MyProxy with a WS-Trust service for passing credentials to clients.

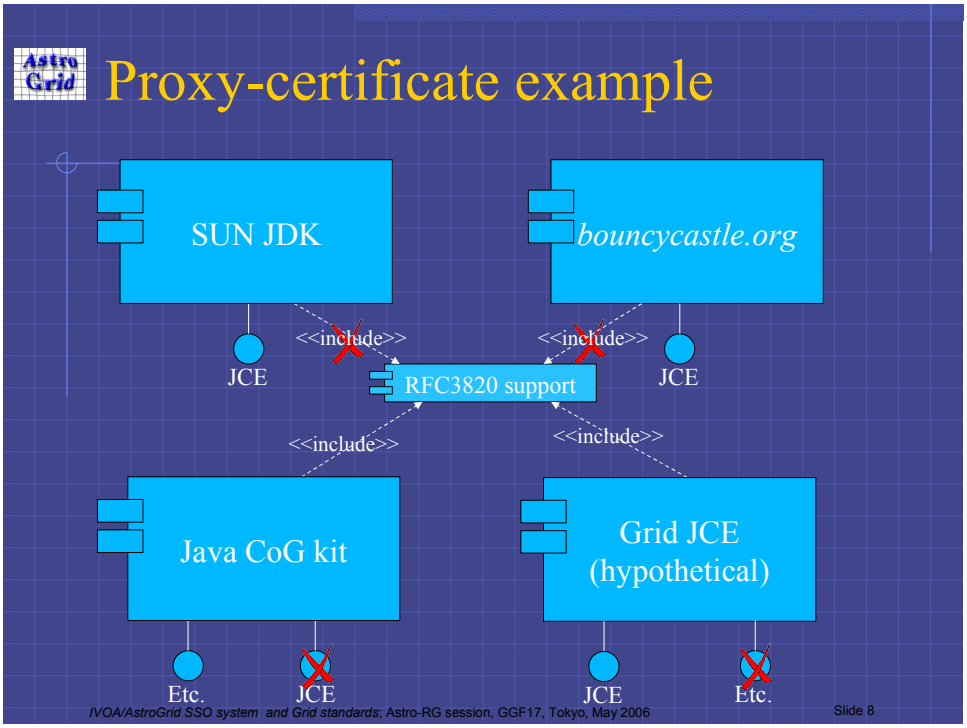


Why did it take so long (Spring 2005 until Spring 2006) to get a prototype working when all the parts are implemented in OSS Grid code?

As noted above, our system isn't implemented "within" a grid toolkit (other than our own). Currently, there is almost no support for concrete Grid standards in mainstream tool-sets and non-Grid frameworks. We can find implementations of great value in many Grid toolkits – some examples from the security domain are shown – but the implementation is often hard to separate from its environment. Frequently, multiple standards are implemented in one large component, leading us to package more classes than we need in our applications. Sometimes a feature seems neatly packaged and reusable, but is actually coupled to its native framework; or to a large collection of 3rd-party code; or to other deployment requirements (e.g. account configuration, naming scheme, permissions, host configuration) that are so natural to the donor toolkit that they are not actually documented.

Equally, mainstream OSS components have the opposite assumption: that grid concepts can be ignored. This is, if anything, even more frustrating than grid-friendly code that we can't easily use. We'd very much like the OSS world to acknowledge at least those grid concepts that have IETF recognition.

We realize that the suppliers of Grid toolkits have no moral obligation to support us with components reusable outside their native framework. However, it may be in the Grid community's interests that they do so. Is the lack of re-usable, loosely-coupled implementation hindering the adoption of the standards? It certainly hinders *our* adoption!



Take RFC3820 – proxy certificates – as an example. We need to work with certificate chains that include proxy certificates. The standard way to do certificates chains in Java is via the Java Cryptography Extension (JCE). *(Press for next effect here.)* There is a JCE implementation bundled in the JDK. Another popular, free one is available from *bouncycastle.org*. Neither of these support proxy certificates. JCoG knows proxies, but is not a JCE provider; it uses a different framework. Therefore, we have to use a non-standard approach in our application code. When other providers finally do support RFC3820, we'll have to change out code back to JCE form to use their products. (*Press for next effect here*.) What we'd really like is a JCE implementation with RFC3820 separate from all the other It would a really good way to "sell" this Grid standard.