Basic Execution Management Services (BES): An Approach

Karl Czajkowski, Ian Foster, and Steve Tuecke for the Globus Alliance

karlcz@univa.com, foster@mcs.anl.gov, tuecke@univa.com

Draft of 9 March 2005

This document catalogs and briefly summarizes the issues of concern to the Basic Execution Management System (BES) discussion from the point of view of the Globus Alliance and the Globus Toolkit architecture, using where possible the terminology set forth in the "Basic EMS Service" draft from Grimshaw, Morgan, Smith, Newhouse, and McCough.

Globus Alliance Position: BES is a straightforward generalization of HPC job submission as seen in existing Grid projects. Today's batch scheduling algorithms are simply implementation components for job managers tailored to enforcing a particular range of operating policies. However, the underlying protocol requirements for Actor creation, monitoring, and management is the same in BES and job submission.

Scoping: We endorse the idea that BES should be generalized enough to support a range of Actor types, from bootable OS images, to executable binaries, Java classes with a Main( ) method, or deployable Web Services.

Patterns: We endorse the notion of Service Containers to which requests to create an Actor are directed. However, we question whether legacy Actors should be treated specially, or whether the creation should always generate what the BES document refers to as a Proxy Actor and which we would refer to as an Actor Management Interface. Whether the Actor itself surfaces as another Web Service is then an application-specific detail of which BES need not concern itself.

Agreement semantics: We believe that the Actor creation pattern is best thought of as a simple agreement-establishment mechanism: after the interaction, the client and Service Container have agreed that the provider will host the Actor (or not). In order to be effectively automated, this interaction must precisely express the nature of the executive instance *and* the nature of the hosting agreement, e.g., it must distinguish between whether the Container is agreeing to satisfy any embedded quality of service constraints or merely agreeing to consider the execution later. In other words, it must distinguish whether the Container has performed all his policy evaluation and “run his scheduler” before accepting the request, or that happen later, in which case the Container is merely creating a provisional Proxy Actor through which the ultimate decision will be visible.

WS-Agreement: There is considerable confusion within GGF about the features and intent of WS-Agreement. As architects and contributors to the WS-Agreement specification, we firmly believe that WS-Agreement is (or will be, upon completion) a good candidate for representing the “Actor creation” pattern referred to above. Still, other

BES message exchanges may be required is to represent the management operations that may target an existing Actor.

Advertisement: In combination with the need for precise agreement semantics to be determined through the creation document exchange, we believe that a long-term viable BES model requires that Containers export a precise advertisement of their “effective capabilities”. The effective capabilities are the range of supported Actor Instance Descriptors, constrained by the parameterization limits of the provider’s resource pool as well as by the operating policies of the provider. The problem here is that an automated client must eventually be able to compare (structurally) the Actor Instance Descriptor for the Actor he wishes to create against the advertisements of providers in order to select a viable Service Container.

Uncertainty: Due to the practical inability for advertisement to precisely target an individual consumer, the advertisement may not encode all of the parameterization constraints caused by policies affecting a particular consumer. This, coupled with the intrinsically dynamic nature of the Grid, means that there remains an uncertainty in whether agreement can be made, no matter how perfectly the advertisement matches the client’s goals. In other words, the agreement step itself is the authoritative arbitration step in BES, while advertisement and discovery are at best hints for optimization as compared to some random trial-and-error strategy.

Discovery: The prerequisite for automated Actor placement is a discovery system that allows clients to find relevant Containers. Clearly, we must assume an externally provided discovery system, e.g., distributed registries or directories of services. However, BES advertisements should be applicable in such systems. Exposing advertisements as BES service metadata is a good basis upon which to hook BES into a discovery framework.

WSRF: As accompanying presentation of the Globus Toolkit 4.0 GRAM illustrates, the use of WSRF modeling conventions and implementation tooling can greatly simplify the rendering of BES-like interfaces. In particular, WS-Addressing mechanisms can be used to provide transportable reference to a specific Actor, and WS-ResourceProperties mechanisms can be used to model a domain-extensible set of Annotations for communicating the status of the Actor. Additionally, WS-BaseNotification can be useful to render the idiom of asynchronous transmission of dynamic Annotations to the client or another interested third party.

Bottom-up: To be widely useful, BES must be designed “bottom-up” to provide management of the range of concrete and more abstract execution systems. The concrete capabilities are rather well understood, and the main exercise is to generalize them for appropriate scoping, without damaging applicability to current systems or future abstractions. For example, a lower tier of BES systems will only understand concrete Application Instance Descriptors in terms of binary files, data, and startup parameters and will have no use for abstract Application Descriptors. At the same time, even these lower tier components should support the association of Annotations with an Actor, even when those Annotations are asserted and only understood by a higher-level component that is managing the Actor.

Time is of the essence: While there are many difficult challenges facing the development of a BES model that will stand the test of time, there is a pressing need for BES interoperability standards now. Any work group activity must strive to produce conservative, basic works that can be applied to a core set of interoperability problems today. More complex or contentious problems should be addressed through composition of BES with other domain-specific content and services and/or extension or refinement of the document content standards used in BES messages. Depending on the target schedule for BES, we would advocate rendering a simple creation WSDL interface or waiting for WS-Agreement to be finalized so that it can be used for Actor creation.

Static tooling: We believe that a successful Web Service rendering of BES must consist of a relatively small number of PortTypes which are generic enough to be shared by all (or nearly all) deployments of BES. Extensibility should happen within the document-oriented payloads of the operations grouped in these PortTypes, rather than through the introduction of myriad new PortTypes over time. New PortTypes should be considered only for "generational" advancement or revision of the BES or EMS model, e.g. transition from an ad-hoc WSDL to use WS-Agreement instead, rather than to support typical scenarios for Actor or Container heterogeneity.

Protocol vs. API: We believe that the role of standards for fundamental technologies like BES should be to provide protocol interoperability. While APIs are important for application work, the main purpose of BES is to enable construction of other brokers and middleware---protocol-level standards are needed to make sure these different proprietary and open source components can function together in the Grid. The Web services architecture is meant to provide such protocol standards, and only indirectly implies APIs through the tooling one uses to generate "stubs" for the message exchanges. It is important that the BES activity keep in mind that we are developing a message exchange protocol and not an API.

Binding Independence: A recurring problem in GGF discussion involving Web Services is what kind of binding technologies are assumed. The Web services architecture, in theory, assumes binding neutrality and focuses on using WSDL for abstract application protocol modeling. This means that issues like reliability and performance are meant to be addressed orthogonally by binding mechanisms and tooling-specific extensions. Practical systems that use BES will need to concern themselves with issues beyond the WSDL itself, but we think these "profiles" must emerge from community practice after BES itself is deployed. The only binding-level concerns we should consider are questions of whether certain required behaviors can be achieved in more than one way with current tools, i.e., to evaluate whether EMS protocol proposals are too restrictive in terms of what bindings are feasible for use with EMS messages.

Automation via extension: The single most difficult challenge for BES and EMS is to provide viable support for automated clients, brokers, meta-schedulers, etc., which can function in an environment with multiple BES Service Containers, each supporting different subsets of executive type, environment parameterization, quality of service levels, and differentiated policy. Systems to date fail to support realistically federated, i.e. Grid, levels of automation because they require human-level intelligence and/or significant out-of-band conventions to plan around and distinguish different providers and execution instances. These systems fail because of poor information modeling. At best, many wasted operations may be attempted to discover (too late) which capabilities are available from a provider; at worst, faults due to unsupported capability requirements may not be automatically distinguishable from other kinds of runtime error. We do not believe this problem can be solved in an initial BES standard any more than it has been solved in the many existing Grid job management systems. This problem is very difficult and cannot be made a requirement for BES to solve initially.