



Update on the WS-Agreement

2005/10/03 GRAAP-WG



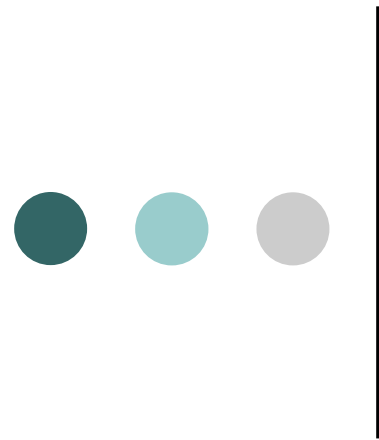
GGF Full Copyright Notice

Copyright (C) Global Grid Forum (2005). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the GGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the GGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the GGF or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE GLOBAL GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."



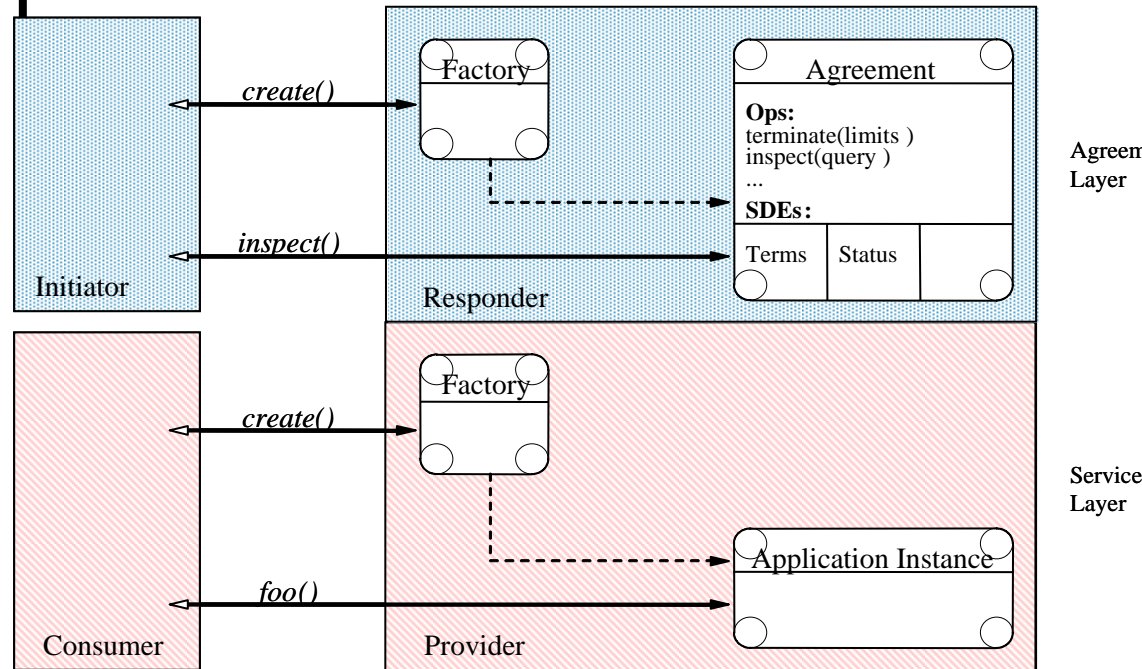
Just a Brief Overview



Overview of WS-Agreement

- An agreement defines a dynamically-established and dynamically-managed relationship between **(two)** parties. The object of the relationship is the **delivery of a service** by one of the parties within the context of the agreement. The management of this delivery is achieved by agreeing on the respective roles, rights and obligations of the parties. The agreement may specify not only functional properties for identification or creation of the service, but also non-functional properties of the service such as performance or availability.
 - Entities can dynamically establish and manage agreements via Web service interfaces.
- WS-Agreement itself is sort of a frame-work for getting agreements on several different domains. The details of things to be agreed are of course domain-specific and is out of the scope of the WS-Agreement Spec.
 - Eg. Job-submission using JSDL is a candidate to be used within the frame-work.
- Relation to other specification
 - Relies on WS-Addressing, WS-ResourceProperties, WS-ResourceLifetime and WS-Base Faults

A Two Layered Model



Agreement Layer: Provides a Web service-based interface that can be used to create, represent and monitor agreements with respect to provisioning of services implemented in the service

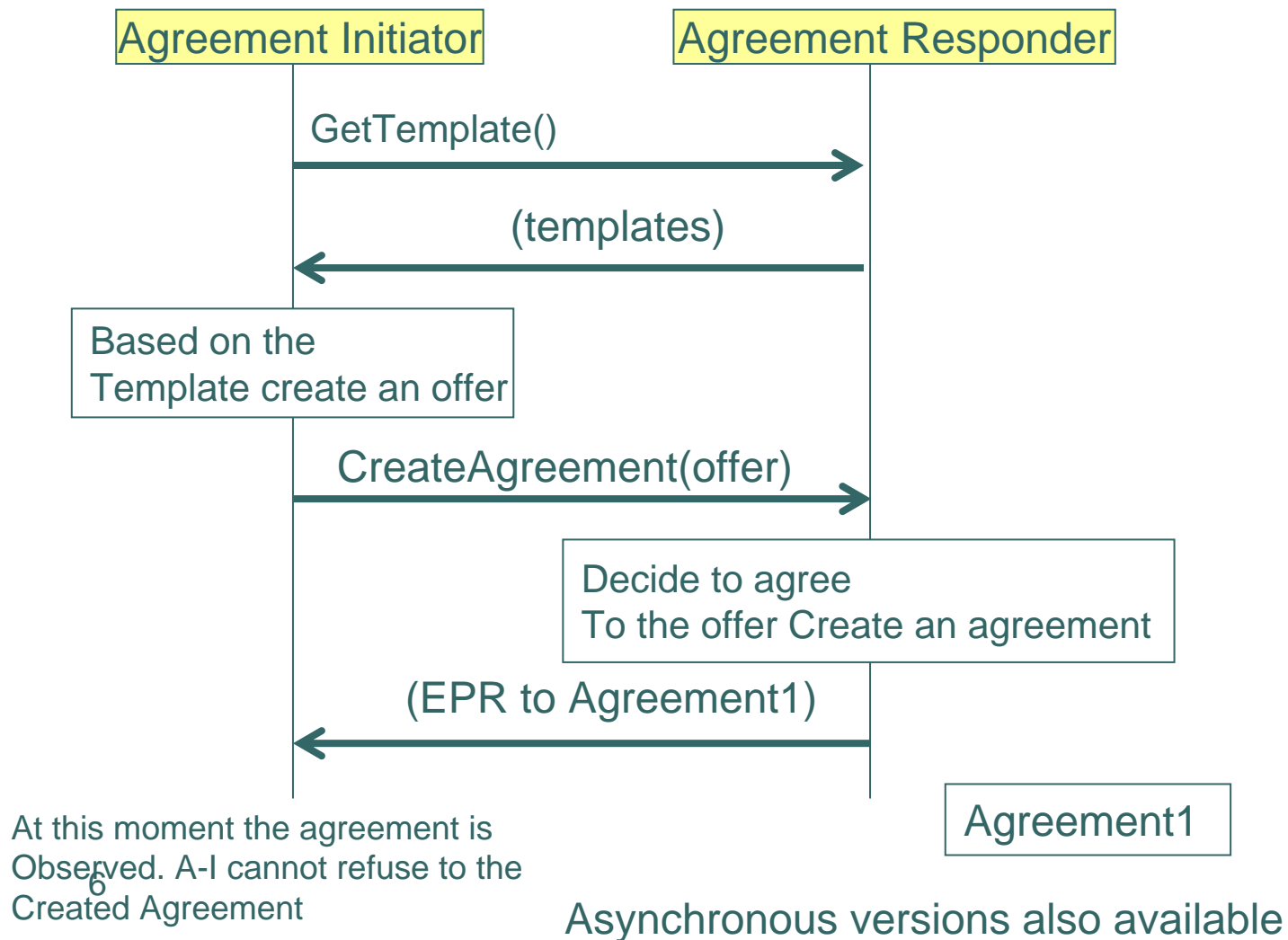
Agreement Layer

The service layer represents the application-specific layer of the service being provided. The class of provided service MAY or MAY NOT be a Web service interface.

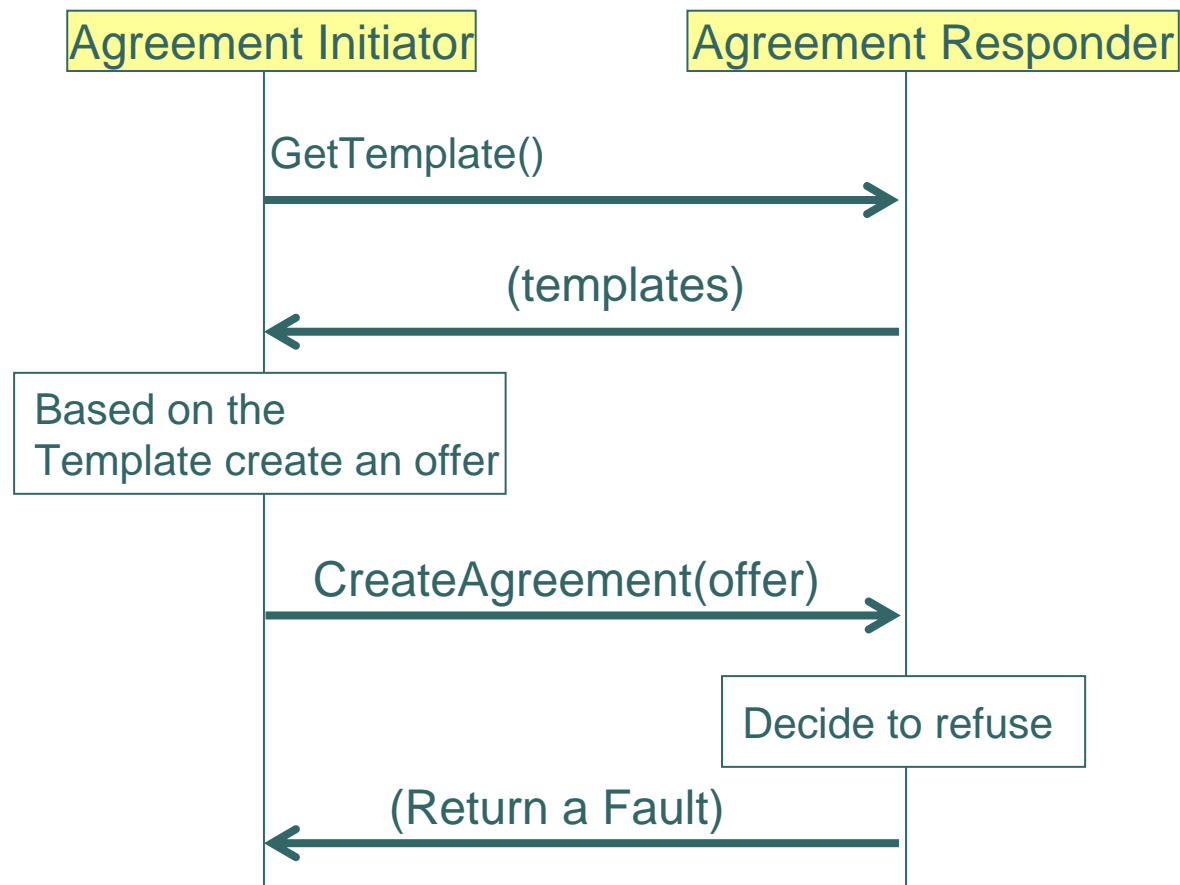
Service Layer

- Whether Agreement Initiator becomes a Service Consumer or Service Provider (ie Agreement Responder becomes a Service Provider or service Consumer) is completely domain dependent

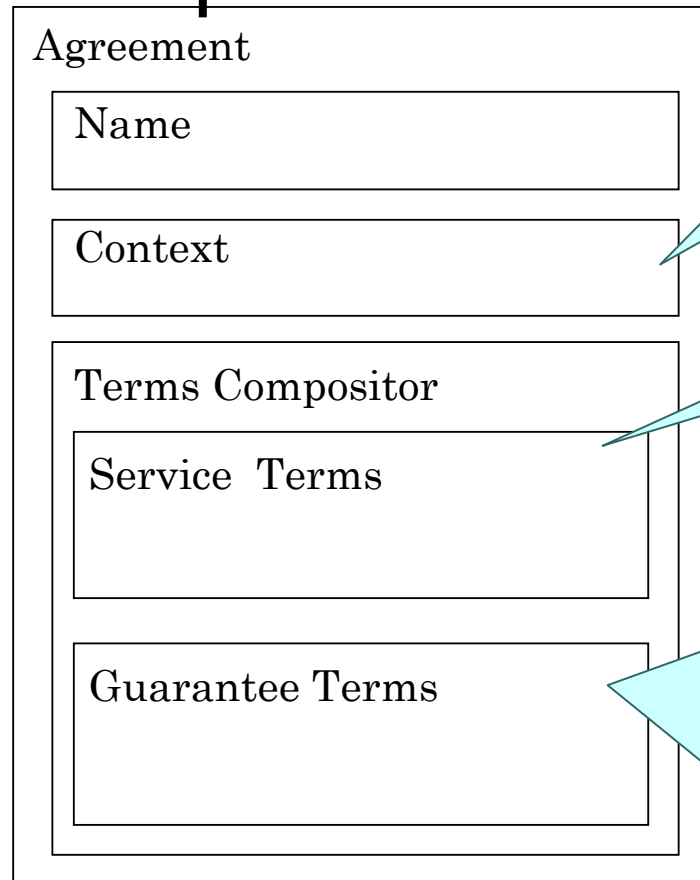
Simple Sequence-1



Simple Sequence-2



Agreement Structure



Information about the Agreement itself

- AgreementInitiator
- AgreementResponder
- ExpirationTime
- . . .

Information about the Service Itself

- Contents are Domain Dependent
- Eg.: Job Description(Program name, Number of nodes etc)

Information about Service Levels which should be Guaranteed

- QualifyingCondition(An optional condition that must be met (when specified) for a guarantee to be enforced. Eg: Timespan when the requests can be submitted:Weekdays, etc)
- ServiceLevelObjective(Conditionthe condition that must be met to satisfy the guarantee. Eg:Needs 128 MB of memory..)
- . . .

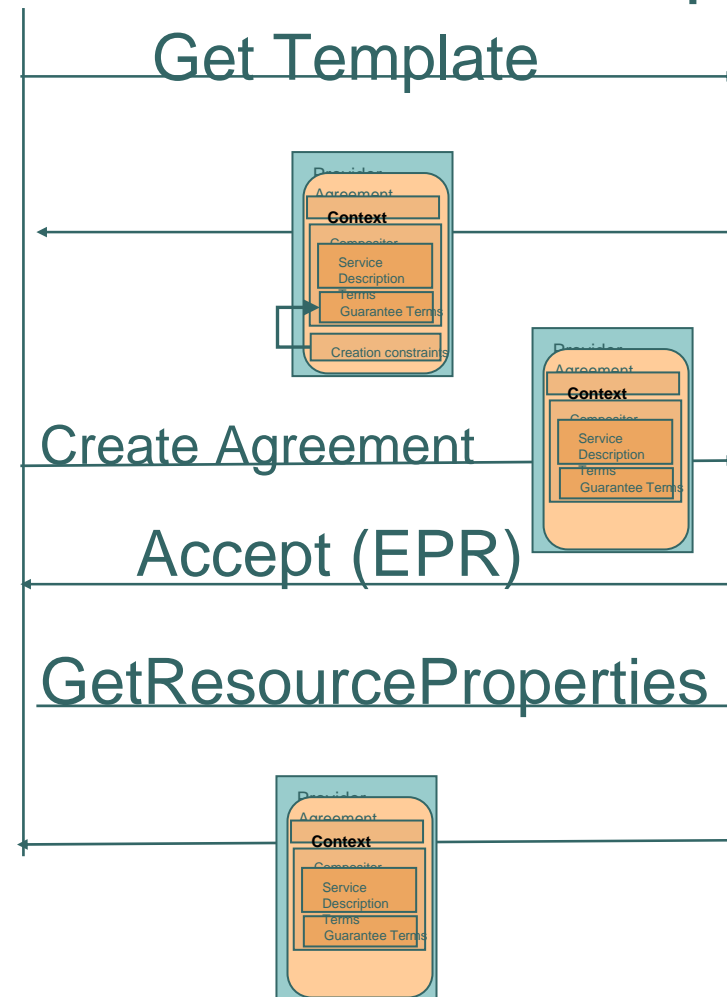
- The structure of an **Agreement Template** is the same as that of an agreement, but an Agreement template MAY also contain a **creation constraint** section, i.e. a section with constraints on possible values of terms for creating an agreement. The constraints make it possible to specify the valid ranges or distinct values that the terms may take.

Typical Agreement Lifecycle

- Four phases in the lifecycle:
 - Exploration: a service provides templates describing possible agreement parameters
 - Creation: Consumer fills in parameters, and makes an offer
 - Operation: Agreement state available as a ResourceProperty
 - Termination: Agreement destroyed explicitly or via soft state (termination time)

Initiator

Responder





What's new in the current document

- Clear Separation of Agreement Initiator/ Agreement Responder vs. Service Consumer/ServiceProvider
- Obligated attribute in the Guarantee terms
- CreatePending Agreement
- Service State made extensible
- Choices in Creation Constraints

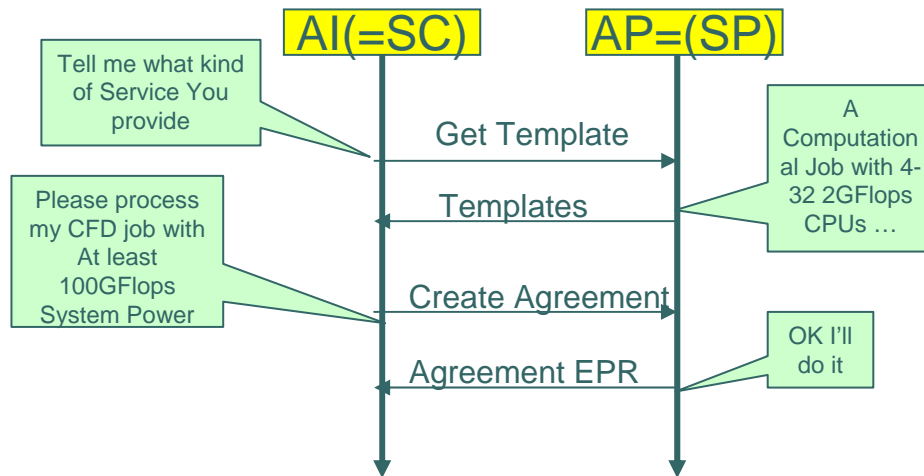
- --Not a Spec change but example change--
- Use JSDL Description rather than the job domain where possible

Clear Separation of Agreement Initiator/ Agreement Responder vs. Service Consumer/Service Provider

- Whether Agreement Initiator (AI) becomes a Service Consumer or Service Provider is completely domain dependent.
- However there always seemed to be a confusion (or assumption) that Agreement Provider(prev. term) = Service Provider

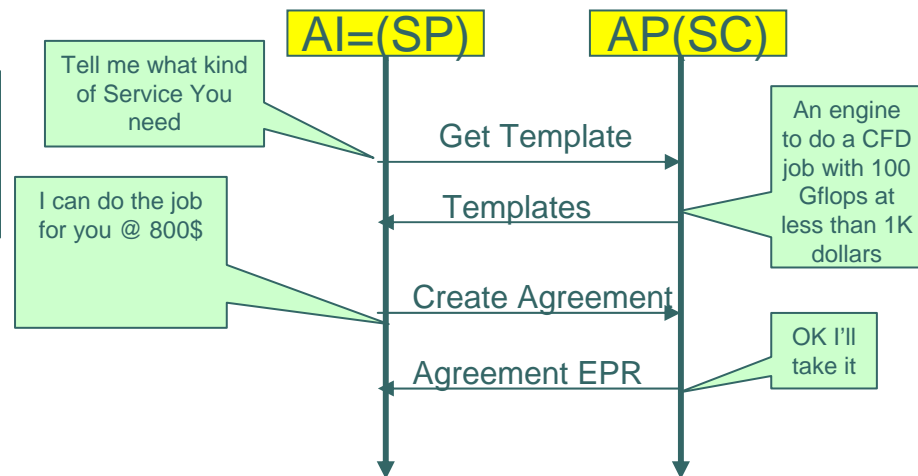
Domain A:

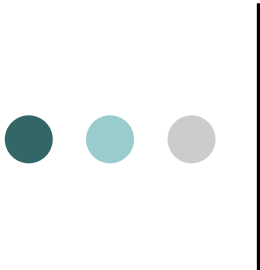
- Initiator: Individual with Job to be run(=Service Consumer)
- Responder: Provider of job execution service(=Service Provider)



Domain B:

- Initiator: Provider of job execution service(=Service Provider)
- Responder: Individual with Job to be run(=Service Consumer)





Clear Separation of Agreement Initiator/ Agreement Responder vs. Service Consumer/Service Provider (Contd.)

- In all the document changed Agreement Provider (Old)=>Agreement Responder
- In the Agreement Context we now have a wsag:ServiceProvider element.

```
<wsag:Context xsd:anyAttribute>
  <wsag:AgreementInitiator>xs:anyType</wsag:AgreementInitiator> +
  <wsag:AgreementResponder>xs:anyType</wsag:AgreementResponder> +
  <wsag:ServiceProvider>wsag:AgreementRoleType</wsag:ServiceProvider>
  <wsag:ExpirationTime>xsd:DateTime</wsag:ExpirationTime> +
  <wsag:TemplateId>xsd:string</wsag:TemplateId>
  <wsag:TemplateName>xsd:string</wsag:TemplateName> ?
  <xsd:any/> *
</wsag:Context>
```

While we're discussing Agreement Context...

This OPTIONAL element refers to the specific version of the template from which this offer or agreement is created. If a template was used to create an offer, the TemplateId in the Context is **MUST be set**.

This element identifies the service provider and is either AgreementInitiator or AgreementResponder. The default is AgreementResponder.



Obligated party in the Guarantee terms

- Previous assumption had been that service provider was the only party responsible for guaranteeing service qualities.
- However, an agreement MAY require a service consumer to give guarantees if the provider's service depends on it. => Introduction of the Obligated party attribute.

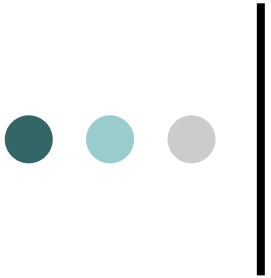
```
<wsag:GuaranteeTerm Obligated="wsag:ServiceRoleType">  
  <wsag:ServiceScope ServiceName="xsd:NCName">  
    xsd:any  
  </wsag:ServiceScope>  
  <wsag:QualifyingCondition>...</wsag:QualifyingCondition>  
  <wsag:ServiceLevelObjective>...</wsag:ServiceLevelObjective>  
  <wsag:BusinessValueList>...</wsag:BusinessValueList>  
</wsag:GuaranteeTerm>
```

Can be either ServiceConsumer or ServiceProvider



Asynchronous version of Create Agreement & Agreement States

- Sometimes, Agreement Responder might take a long time to decide whether it can meet the demands of the Agreement Initiator.
 - Eg. Proxy for a Wide area network based data-centers might take time to determine which of the sites can provide the service.
- =>Introduction of Asynchronous version of createAgreement named createPendingAgreement + Introduction of the Agreement States.
 - Agreement can have the following states:
 - Pending: The Pending state means that an Agreement offer has been made but it has been neither accepted nor rejected
 - Observed: The Observed state means that an Agreement offer has been made and accepted.
 - Rejected. The Rejected state means that an Agreement offer has been made and rejected.
 - Complete. The Complete state means that an Agreement offer has been received and accepted, and that all activities pertaining to the Agreement are finished.

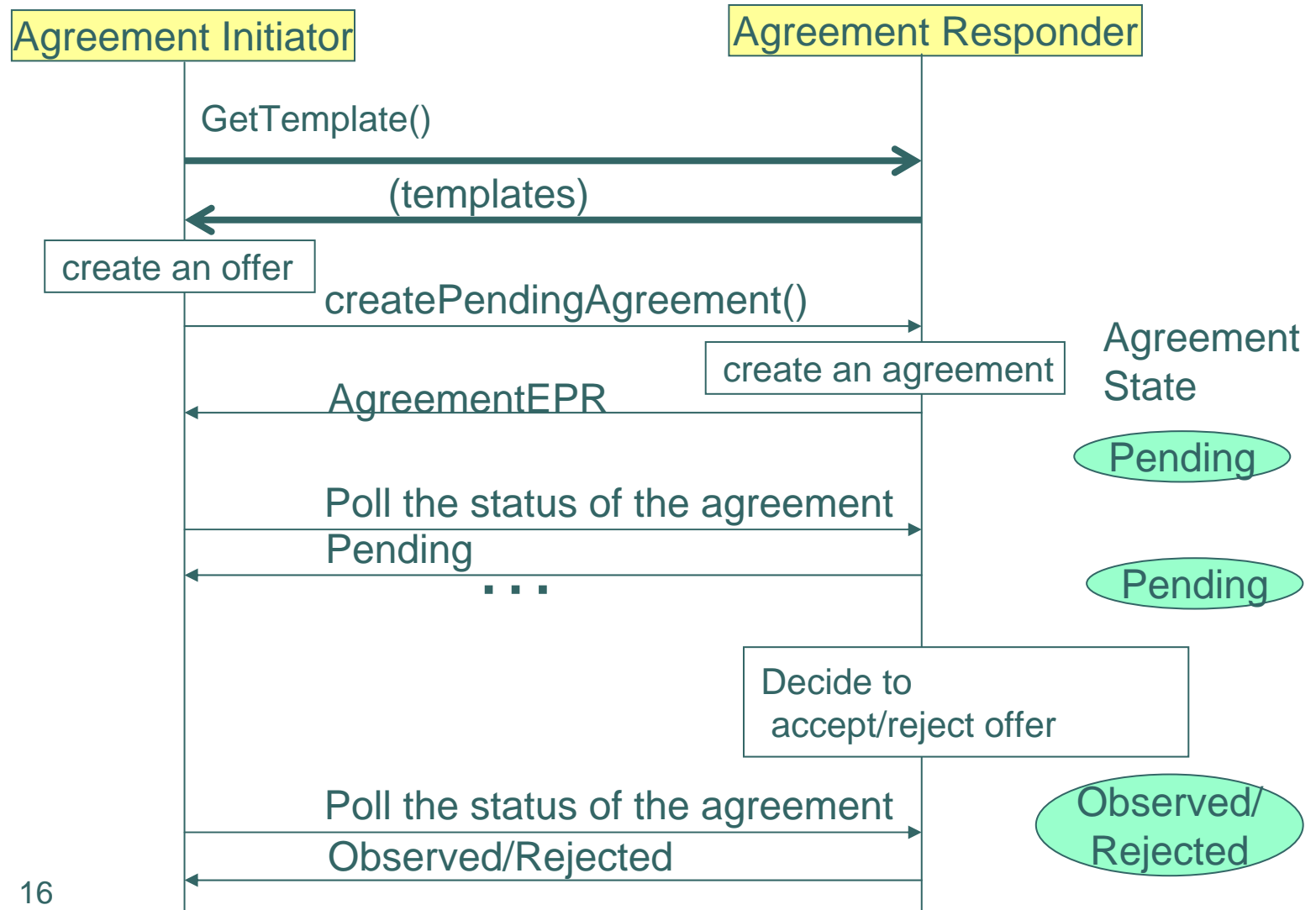


Asynchronous version of Create Agreement & Agreement States (Contd.)

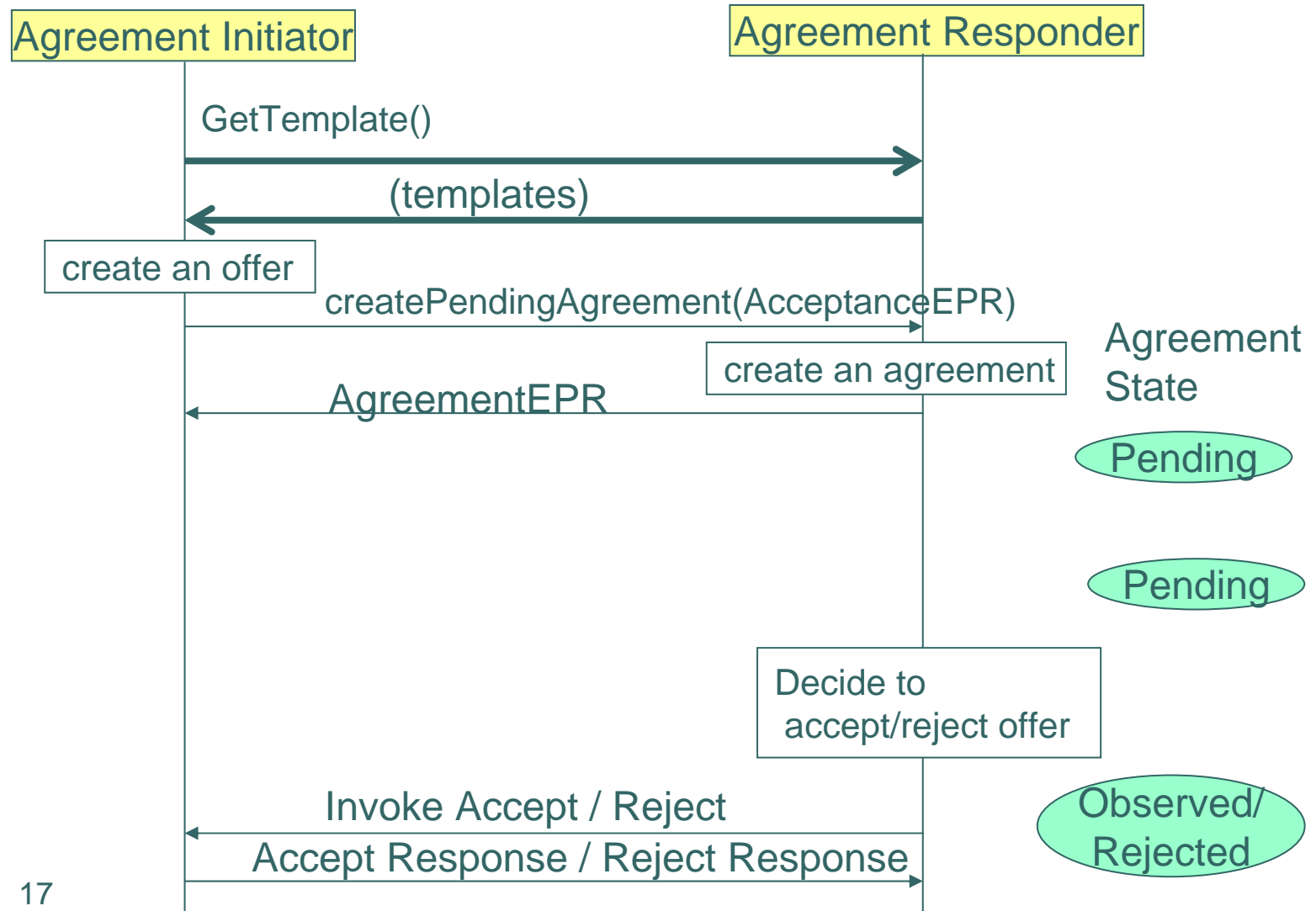
- `createPendingAgreement` will return immediately but the decision for accepting or rejecting the offer might not have been made in which case the Agreement's state is defined as Pending.
- So how can an Agreement Initiator who has issued `createPendingAgreement` know whether the Agreement Responder has agreed to the Agreement or not?
 - By polling the state of the Agreement
 - Or at `createPendingAgreement` time specifying an EPR where `wsag:Accept` or `wsag:Reject` operation will be invoked by the Agreement Responder.

Cf. Next two pages

Sequence Image for Polling

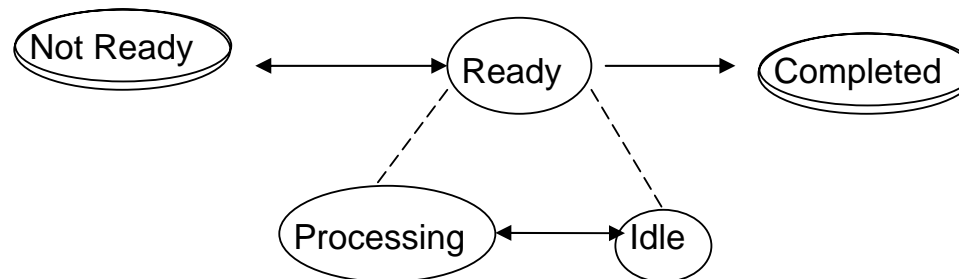


Sequence Image for Accept/Reject



Service State made extensible

- **Not Ready**, **Ready** and **Completed** are the normative primary states of a service description term. Each state can be extended with one or more sub-states in a specific usage domain. **Processing** and **Idle** are two normative sub-states of the primary state **Ready**.
 - **Not Ready** – The service cannot be used yet.
 - **Ready** – The service can start now to be used by a client or to be executed by the service provider.
 - **Processing** – The service is ready and currently processing a request or is otherwise active.
 - **Idle** – The service is ready, however currently not being used.
 - **Completed** – The service cannot be used any more and any service provider activity performing a job is finished.





Choices in creation constraints

- In GGF14, there had been requests for Creation constraints to be able to offer Either serviceA or serviceB or serviceC. (And have the Agreement Initiator specify which service it needs.)
- One candidate had been the term compositors but that might make the implementation too complex
- => Allow the use of `xsd:typeDefParticle`

```
<wsag:Item Name="xsd:NCName">
  <wsag:Location>
    xsd:anyType
  </wsag:Location>
  <wsag:ItemConstraint>
    <xsd:restriction>
      xsd:simpleRestrictionModel
    </xsd:restriction> ?
    <xsd:group>xs:groupRef</xsd:group> ?
    <xsd:all>xs:all</xsd:all> ?
    <xsd:choice>xs:explicitGroup</xsd:choice> ?
    <xsd:sequence>xs:explicitGroup</xsd:sequence>?
  </wsag:ItemConstraint>
```

Job Submission Example Using prototype JSDL document as Service terms: Template Example

```
<wsag:ServiceDescriptionTerm
  wsag:Name="Job JSDL" wsag:ServiceName="Job">
  <jSDL:JobDefinition>
    <JobDescription>
      <Application>
        <jSDL-posix:POSIXApplication>
          <FileSizeLimit>1048576</FileSizeLimit>
          <CoreDumpLimit>0</CoreDumpLimit>
          <OpenDescriptorsLimit>64</OpenDescriptorsLimit>
        </jSDL-posix:POSIXApplication>
      </Application>
      <Resources ...>
        <OperatingSystem>
          <OperatingSystemType>
            <OperatingSystemName>LINUX</OperatingSystemName>
          </OperatingSystemType>
        </OperatingSystem>
        <CPUArchitecture>
          <CPUArchitectureName>x86</CPUArchitectureName>
        </CPUArchitecture>
        <IndividualCPUSpeed>
          <Exact>1600000</Exact>
        </IndividualCPUSpeed>
        <IndividualCPUCount>
          <Exact>2.0</Exact>
        </IndividualCPUCount>
        <IndividualNetworkBandwidth>
          <Exact>100000000</Exact>
        </IndividualNetworkBandwidth>
        <TotalResourceCount>
          <Exact>1</Exact>
        </TotalResourceCount>
      </Resources>
    </JobDescription>
  </jSDL:JobDefinition>
</wsag:ServiceDescriptionTerm>
```

•Default 1 MB file size limit

•Default 0 byte core dump size limit

•Default 64 open file descriptors limit

•Default "LINUX" operating system

•Default "x86" CPU type

•Default 1.6 GHz CPU speed

•Default 2 CPUs per node

•Default 100 Mb/s network connectivity for nodes

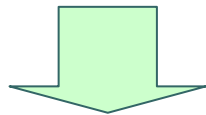
•Default 1 node per job

Job Submission Example Using prototype JSDL document as Service terms contd.

- Maximum 500 MB file size limit (Default 1MB)
- Maximum 500 MB core dump size limit (Default 0MB)
- Maximum 1024 open file descriptors limit(Default 64)

```
<jSDL:JobDefinition>
  <JobDescription>
    <Application>
      <jSDL-posix:POSIXApplication>
        <FileSizeLimit>1048576</FileSizeLimit>
        <CoreDumpLimit>0</CoreDumpLimit>
        <OpenDescriptorsLimit>64</OpenDescriptorsLimit>
      </jSDL-posix:POSIXApplication>
    </Application>
  </JobDescription>
</jSDL:JobDefinition>
```

Template:Service Description Term



```
<FileSizeLimit>16777216</FileSizeLimit>
<CoreDumpLimit>0</CoreDumpLimit>
<OpenDescriptorsLimit>1024</OpenDescriptorsLimit>
```

Offer: Service Description Term

```
<wsag:Item>
  <Location>//jSDL-posix:FileSizeLimit</Location>
  <xsd:restriction base="xsd:positiveInteger">
    <xsd:maxInclusive value="524288000"/>
  </xsd:restriction>
</wsag:Item>
<wsag:Item>
  <Location>//jSDL-posix:CoreDumpLimit</Location>
  <xsd:restriction base="xsd:positiveInteger">
    <xsd:maxInclusive value="524288000"/>
  </xsd:restriction>
</wsag:Item>
<wsag:Item>
  <Location>//jSDL-posix:OpenDescriptorsLimit</Location>
  <xsd:restriction base="xsd:positiveInteger">
    <xsd:maxInclusive value="1024"/>
  </xsd:restriction>
</wsag:Item>
```

Template: Creation Constraint

- 16MB file size limit
- 0 MB core dump size limit
- 1024 open file descriptors limit

Job Submission Example Using prototype JSDL document as Service terms contd.

- Choice of "LINUX" or "FreeBSD" (exclusive)
- Choice of "x86", "x86_32", or "x86_64" CPU types (exclusive)

```
<OperatingSystem>
  <OperatingSystemType>
    <OperatingSystemName>LINUX</OperatingSystemName>
  </OperatingSystemType>
</OperatingSystem>
<CPUArchitecture>
  <CPUArchitectureName>x86</CPUArchitectureName>
</CPUArchitecture>
```

Template:Service Description Term



```
<OperatingSystem>
  <OperatingSystemType>
    <OperatingSystemName>LINUX</OperatingSystemName>
  </OperatingSystemType>
</OperatingSystem>
<CPUArchitecture>
  <CPUArchitectureName>x86_32</CPUArchitectureName>
</CPUArchitecture>
```

Offer: Service Description Term

```
<wsag:Item>
  <Location>//jsdl:CPUArchitecture/CPUArhitecturename</Location>
  <xsd:restriction base="jsdl:ProcessorArchitectureEnumeration">
    <enumeration value="x86_32"/>
    <enumeration value="x86_64"/>
    <enumeration value="x86"/>
  </xsd:restriction>
</wsag:Item>
<wsag:Item>
  <Location>
    //jsdl:OperatingSystem/jsdl:OperatingSystemType/jsdl:OperatingSystemName
  </Location>
  <restriction base="jsdl:OperatingSystemTypeEnumeration">
    <enumeration value="LINUX"/>
    <enumeration value="FreeBSD"/>
  </restriction>
</wsag:Item>
```

Template: Creation Constraint

- "LINUX"
- "x86_32"



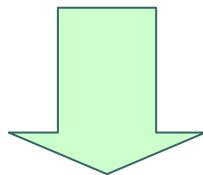
Job Submission Example Using prototype JSDL document as Service terms contd.

- Choice of 10, 100, or 1000 Mb/s network connectivity for nodes
(inclusive)

Template: Creation Constraint

```
<IndividualNetworkBandwidth>  
  <Exact>1000000000</Exact>  
</IndividualNetworkBandwidth>
```

Template:
Service Description Term



```
<wsag:Item>  
  <wsag:Location>//jsdl:IndividualNetworkBandwidth</wsag:Location>  
  <xsd:sequence>  
    <xsd:element name="Exact" minOccurs="1" maxOccurs="unbounded">  
      <xsd:simpleType>  
        <xsd:restriction base="xsd:double">  
          <xsd:enumeration value="1000000000"/>  
          <xsd:enumeration value="1000000000"/>  
          <xsd:enumeration value="1000000000"/>  
        </xsd:restriction>  
      </xsd:simpleType>  
    </xsd:element>  
  </xsd:sequence>  
</wsag:Item>
```

```
<IndividualNetworkBandwidth>  
  <jsdl:Exact>1000000000</jsdl:Exact>  
  <jsdl:Exact>1000000000</jsdl:Exact>  
</IndividualNetworkBandwidth>
```

- Selects 100/1000 Mb/s network speeds
(the scheduler can choose which)