

# Fresco: A Web Services based Framework for Configuring Extensible SLA Management Systems

Christopher Ward, Melissa J. Bucu, Rong N. Chang, Laura Z. Luan, Edward So, Chunqiang Tang  
IBM Thomas J. Watson Research Center  
19 Skyline Drive, Hawthorne, NY 10532  
{cw1, mjbucu, rong, luan, edwardso, ctang}@us.ibm.com

## Abstract

*A Service Level Agreement (SLA) is a service contract that includes the evaluation criteria for agreed service quality standards. Since agreeable specifications on the evaluation criteria cannot be limited in practice, competitive SLA management products must be extensible in terms of their support for contract-specific SLA compliance evaluations. While the need of running and managing those software products as services increases, we have found that developing a good solution for configuring them as per contractual terms is a challenging task. This paper presents the Fresco framework, which facilitates configuring extensible SLA management systems using Web Services. An XML-based specification of SLA management related data called SCOL will also be presented to show how the framework supports contract-specific SLA terms and contract-specific extensions of the deployed SLA management software. The paper furthermore shows how the Fresco system uses a template-based approach to communicate with other Web Services applications with support for various input and output formats. Our experience with implementing the Fresco framework for a leading commercial SLA management software product demonstrates that the framework facilitates the creation of effective and efficient solutions for configuring extensible SLA management systems.*

## 1. Introduction

A service level agreement (SLA) is a contract that includes specifications on a set of agreed service quality management objectives (or service level objectives, SLOs). Details on the specifications (e.g. raw quality measurement data sources, quality measurement data adjudication criteria, service level evaluation rules, SLOs) form part of the rights and obligations between the service provider and the customer [1,2]. An SLA is typically a text document written for humans to interpret and as such the descriptions of these specifications are also described so as to be readily understood by the signing parties. Sample phrases from a real contract for

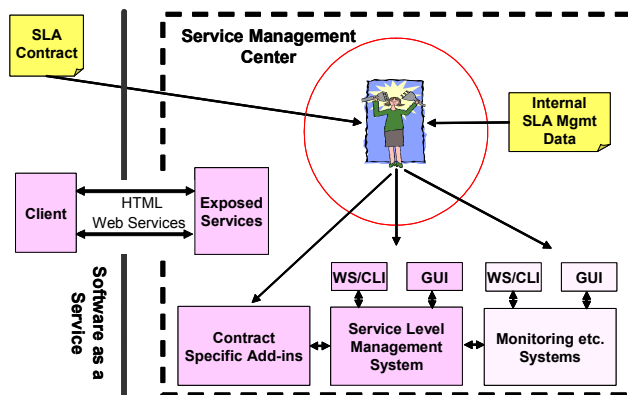
the SLO and adjudication criteria elements are (see Appendix) [3]:

- “Provider’s Service Level Agreement (SLA) standard for Customer’s Web Hosting environment is less than four hours per calendar month of downtime, which is an availability of approximately 99.5%.”; and
- “This SLA objective specifically does not include failures caused by Customer, outages associated with contract maintenance provisions, failure of bandwidth connectivity, and external failures outside the Website Hosting Environment.”

The delivery of *Software as a Service* (SaaS) to customers is already readily available [4] as is the delivery of monitoring data using the Web (e.g. via HTML) [5]. However, as shown in Figure 1, in addition to the offered service itself (e.g. SAP’s Adaptive Computing Concept) a complex service management infrastructure is required to support SaaS. Based on our experience with IBM’s Universal Management Infrastructure (UMI) [4], this is an integration of many commercial systems arranged along functional lines (e.g. monitoring, metering, billing, and SLA management) and maintained by service personnel. From the customers view however, the implementation is irrelevant provided the offered service complies with the quality assurance guarantees as expressed in the SLA contract and reflected in the SLA management system reports. Although competitive SLA management products are necessarily extensible in terms of their support of contract specific SLA compliance evaluations, we have found that *configuring* these systems per contractual terms and conditions remains a challenging undertaking. In other words, configuring existing SLA management systems using Web Services for SLA contracts (including Web Services based SLA contracts) can be a messy business.

There are several issues that make the SLA management domain particularly challenging: (1) the wide variety of SLA compliance evaluation terms necessarily requires the system management tools be

extensible (either using plug-in modules or programmable expressions, or both); and (2) the significant disparity between the information content as expressed in the contractual agreement between the parties and the details required to realize the contract. Given this, as illustrated in Figure 1, it is presently necessary for the service management personnel to “bridge the gap” between a service specification and its realization. That is, they presently configure the SLA management systems in accordance with the contract terms manually through Web Services, via CLI configuration scripts or using the product’s GUI, and perhaps selecting and installing code modules also.



**Figure 1. Service personnel are required to “bridge the gap” between SLA contract data and SLA management system configuration data, a challenging task.**

Such configuration typically requires that the personnel create product specific management system entities, e.g. “Schedules”, “Offerings”, and “Orders”, and perhaps associate them with selected evaluation algorithms during this process. Furthermore, while the SLA contract necessarily provides expressive phraseology directed to satisfying the customers service requirements, the internal SLA management system may not. E.g. the contract may specify “Availability of 99.0% during STANDARD period 24x7, and (an overlapping) 99.9% during PEAK period 9:00-5:00 M-F”. The SLA management system in use however, may support only non-overlapping (or “single-state”) schedules, which enforces every quality measurement data for an SLA be associated with only one defined scheduling state. The service personnel again have to manually “bridge the gap” by creating management instances in such a way as to generate reports that adequately cover the stated quality measurement clauses. Regardless of the transformation selected, the service configuration data and mappings must be maintained within the service management center and form part of a configuration management data base (CMDB).

This paper proposes *Fresco*, a Web Services based framework for configuring SLA management systems to address many of the aforementioned service configuration “pain points”. *Fresco* provides a transformational capability for the service personnel, a configuration tool and a data repository that relate the original SLA specification to the underlying system. Specifically, *Fresco* addresses several challenging issues:

1. how to design a data model that is sufficiently flexible and extensible so that it readily accommodates all the data elements found in real SLA’s and is also sufficiently expressive to represent the management system elements
2. how to represent and manage the association between these SLA contract data and the corresponding configuration data of the flexible SLA management systems
3. how to reduce the complexity for the service management personnel so they are not required to fully understand these relationships

The remainder of this paper is organized as follows. Section 2 provides additional details on the challenges in mapping an SLA contract to the management systems. Section 3 describes *Fresco*, our SLA Management Configurator that addresses the issues raised in Section 2. Section 4 provides a simple illustrative example of the benefit of *Fresco* by showing how it is able to associate service oriented contractually defined multi-state schedule clauses to the management system’s single state schedules. Section 5 discusses some of the related work in this area and we conclude the paper in Section 6.

## 2. The SLA Configuration Challenge

Distinct from our earlier work on the functional specification of a web service based *SLA compliance evaluation service* for UMI, the specific issue addressed in this research is a Web Service based framework to support the management and exploitation of on demand service oriented contractual SLA specifications and the SLA management system product configuration data. As mentioned earlier, there will in general be a schema mismatch between the precise details of the contractually agreed upon terms (which is necessarily the result from customer’s specific needs for the on demand service) and the exploitation and integration of products used to support the contract. This mismatch results in two sets of data elements: (1) SLA contract data elements from a contractual specification perspective; and (2) related management system data elements. Some design drivers for the representation and association of these elements are illustrated by the following points:

1. **Fresco should represent and expose all key SLA data elements.** The SLA management configuration language should reflect all elements

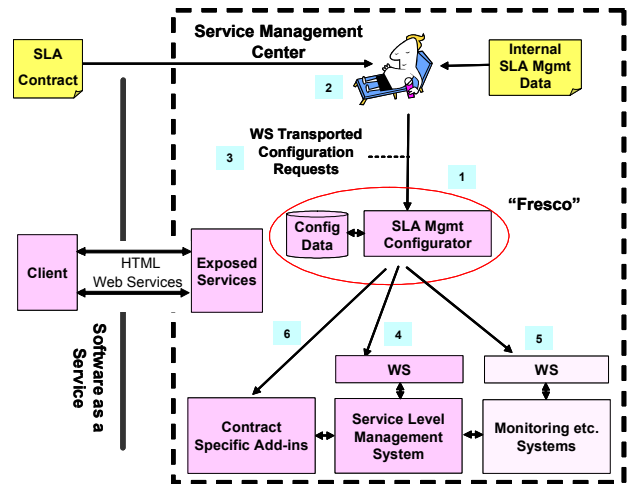
that pertain to SLA contracts. For example, contracts may include clauses that require the time zone of the resource be identified and also include “local” time considerations. E.g.: “PEAK time is 9:00am to 5:00pm *local time*, the maintenance period is to occur at 2:00AM EST”. Such clauses require geographic location be attributable to each resource ID, regardless of whether the underlying management system represents this. Correct interpretation would be a configuration issue – for the configurator. Contracts may also include adjudication clauses that cannot be satisfied by simple state-based data inclusion/exclusion rules, e.g. “failures caused by customer”.

2. **Fresco should allow extensibility in data element representation.** For example, an SLO specification may include an overlapping schedule which is an extension to simpler non-overlapping schedule. E.g.: “Availability of 99.0% during STANDARD period 24x7, and (an overlapping) 99.9% during PEAK period 9:00-5:00 M-F”. Such overlapping schedule specifications should be readily described, even though they may not be easily handled by the management systems notion of a schedule.
3. **Fresco should permit specification of not only SLA data elements but also plug-ins and algorithms.** In support of extensibility, many system management products provide configuration of not only parameter values but also plug-ins. For example, contracts may include SLO clauses that require several concurrent measurement streams in order to perform a computation. E.g.: “99.9 percent of email response time shall be less than 2 seconds when the number of concurrent email users is less than 50 and shall be less than 3 seconds otherwise.” Frequently these specialized clauses require either specifying the evaluation formula or loading customized metric evaluators which must be separately and carefully configured.
4. **Fresco should support flexible data element sharing between contracts.** Often the exact details of data sharing (or commonality) between “similar” contracts depends on the nature of the offering and associated SLAs. This introduces flexibility requirements for configuration which may differ from ???commonality model assumption for the underlying management system.

### 3. The Fresco Framework

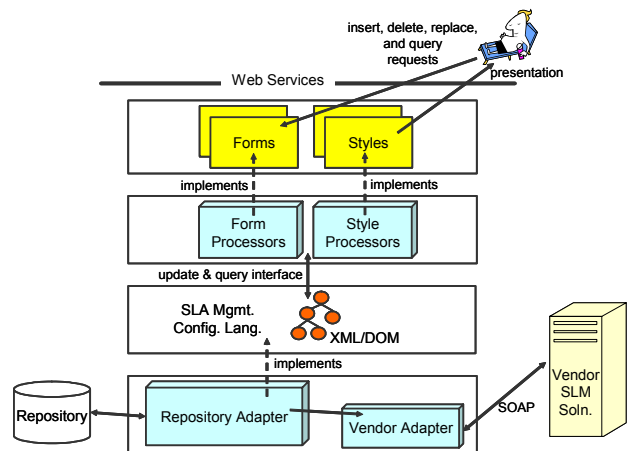
A functional overview of the Fresco configurator is provided in Figure 2. Referring to the numeric boxes: Fresco (1) supports the service personnel who receive SLA contract deployment requests (2) and associated SLA management (SLM) data. These are used to issue,

through appropriate interfaces, Web Service requests (3) to Fresco for the new SLA. Fresco then configures necessary SLA management (4) and other systems (5) including any necessary plug-ins as needed (6).



**Figure 2. Fresco strives to manage the complexity of SLA management configuration through SLM data relationship management.**

The system architecture for Fresco is shown in Figure 3. It leverages an XML based SLA configuration language that we have developed called SCOL (detailed in Section 3.1), makes use of Form and Style processors (detailed in Section 3.2), and is accessed through a template driven client interface (detailed in Section 3.3). Fresco also includes vendor adaptors that are responsible for emitting vendor relevant commands (either via Web Services or CLI) for appropriate SLA elements. The adaptor pattern is used so that the system can easily bind to different vendor products and easily accommodate change.



**Figure 3. Fresco primary functional elements.**

Fresco is implemented as a J2EE 1.3 Web Application (WAR) running in IBM WebSphere Application Server

Version 5.1. It interacts with a leading commercial management system product through a Web Services based adaptor.

SCOL is exposed through a repository adaptor (lower left of Figure 3) as a XML repository and may be considered part of a logical CMDB. Either a native XML store or XML surfaced on a relational database could have been used. We have experimented with both and presently exploit the relational model using IBM DB2 Universal Database Enterprise Server Edition Version 8.2. For now, the relational model better supports updates to the data store and permits efficient realization of certain types of high level API queries (see Section 3.3). We would expect that native XML stores with full schema support for updates will fully support our repository design in the future.

### 3.1 XML Specification for SLA Relationships

At the heart of our proposed solution is the effective representation of SLA/configuration data. We introduce an SLA management configuration language called SCOL that represents both the data elements for an abstract representation of an SLA contract and also SLM system configuration information. The abstract representation is based on the same semantic elements identified in [6] and by DIRECT [7]. The actual vocabulary is derived from real IT service contracts, and has been validated by an extensive analysis of over 75 contracts. A sampling of some top level elements is provided in Figure 4. For flexibility and extensibility the entire language is represented in XML and leverages XML's hierarchical structure and its ease in representing non-hierarchical relationships using XPath expressions.

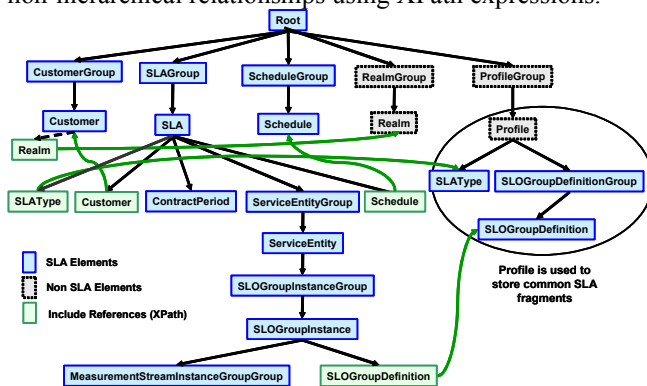


Figure 4. Illustrative top level elements in the SLA Management Configuration Language (SCOL).

As can be seen from the figure the “complete” data model is represented under a `Root` element with major top level grouping corresponding to primary SLA contract entities. The illustrative fragment has five top level groups: `Customer`, `SLA`, `Schedule`, `Realm` and `Profile`. Each group provides instance information for that schema element, e.g. an `SLA` is typed and is

comprised of a `Customer`, `ContractPeriod`, a group of `ServiceEntity` elements and reference to various `Schedule` elements.

Extensibility of the XML specification is demonstrated in the illustrative XML for sample elements `Customer` and `Schedule` in Figure 5 and `SLA` in Figure 6.

```

- <CustomerGroup>
- <Customer>
  <Name primary="Acme Corporation" id="8989" />
  <Description />
  <ExtendedData />
- <RealmGroup>
  <Realm includeReference="store:/Root/RealmGroup/Realm[Name
    [@primary="ACME"]]" />
  </RealmGroup>
</ExtendedData>
</Customer>
</CustomerGroup>
- <ScheduleGroup>
- <Schedule type="NonoverlappingFirstStateIsUsed">
  <Name primary="StandardAndPeak" id="2122" />
- <StateGroup>
  <State>CRITICAL</State>
  <State>PEAK</State>
  <State>PRIME</State>
  <State>STANDARD</State>
  <State>LOW_IMPACT</State>
  <State>OFF_HOURS</State>
  <State>NO_SERVICE</State>
</StateGroup>
  <DefaultState>NO_SERVICE</DefaultState>
  <IsAuxiliary>false</IsAuxiliary>
  <AuxiliaryScheduleGroup />
- <PeriodGroupGroup>
+ <PeriodGroup state="NO_SERVICE">
+ <PeriodGroup state="PEAK">
- <PeriodGroup state="STANDARD">
  - <Period>
    <PeriodStart time="00:00:00" timezone="America/New_York" />
    <PeriodEnd time="23:59:59" timezone="America/New_York" />
  - <Frequency>
    <Daily />
  </Frequency>
  </Period>
</PeriodGroup>
</PeriodGroupGroup>
</Schedule>
</ScheduleGroup>

```

Figure 5. Customer and Schedule Specifications.

Notice that each SLA management element is identified by a Name and includes certain required fields. These were selected based on reviewing contracts and management systems. E.g., the schedule allows for the specification of state names (within `StateGroup`), periods (within `PeriodGroup`) etc., and also corresponds directly with the semantics of at least one SLA management system product. Certain elements may also contain additional (`ExtendedData`) data, as in the case of the `Customer` above. Profiles are used to represent commonalities allowing new instances to be created by cloning and modifying the non-profile elements and “including” the profile. Relationships in SCOL are represented as attributes using a modified XPath expression. Two of the most common relationships are `includeReference` and `relatedToReference`. The former can be considered analogous to a “symbolic link” in a file system in that the element to be included is used, but is annotated with the `includeReference` attribute to reference the actual

element in question and is an excellent way to capture commonalities that can be shared by other entities, e.g. the inclusion of realms with the customer. The latter relationship is used to provide contextual information for elements where needed, e.g. in Figure 6, an SLA instance has a relationship with a particular profile (see Figure 4), that is, draws elements from the profile.

```

- <SLA>
- <ProfileGroup>
  <Profile relatedToReference="store:/Root/ProfileGroup/Profile[Name
    [@primary='ACMEWebOnDemand']]"/>
  </ProfileGroup>
  <Name primary="ACMEWebOnDemand1" id="6757"
    alias="ACMEWebOnDemand1--ACMEWeb Application Server On
    Demand Environment"/>
  <Description />
  <SLAType includeReference="store:/Root/ProfileGroup/Profile[Name
    [@primary='ACMEWebOnDemand']] /SLAType"/>
  <Customer includeReference="store:/Root/CustomerGroup/Customer
    [Name[@primary='Acme Corporation']]"/>
  <Provider />
- <ContractPeriod>
  <StartDate date="2004-07-01" timezone="America/New_York"/>
  <ExpirationDate />
  </ContractPeriod>
- <ServiceEntityGroup>
- <ServiceEntity>
  <Name />
- <SLOGroupInstanceGroup>
- <SLOGroupInstance>
  <SLOGroupDefinition
    includeReference="store:/Root/ProfileGroup/Profile
    [Name
    [@primary='ACMEWebOnDemand']] /ServiceEntityGroup /Serv
    [Name[@primary='IP Host+AvailabilityPercent']]"/>
- <MeasurementStreamInstanceGroupGroup>
- <MeasurementStreamInstanceGroup>
- <MeasurementStreamInstance>
  <MeasurementStreamSpecification
    relatedToReference="store:/Root/SLAGroup/SLA
    [Name
    [@primary='ACMEWebOnDemand']] /ServiceEntityGroup.
    [Name[@primary='IP
    Host+AvailabilityPercent']] /MeasurementStreamSpecifi
    [Name[@primary='DEFAULT']]"/>
  <Name primary="iphost89.abh.company.com"/>
  <Timezone>America/New_York</Timezone>
  </MeasurementStreamInstance>
  + <MeasurementStreamInstance>
  </MeasurementStreamInstanceGroup>
  </MeasurementStreamInstanceGroupGroup>
  </SLOGroupInstance>
  </SLOGroupInstanceGroup>
  </ServiceEntity>
  </ServiceEntityGroup>
  + <InterSLARelationshipGroup>
  </SLA>

```

Figure 6. SLA Contract Specification.

Compare this with the `includeReference` within the `SLOGroupDefinition` that “includes” the common elements associated with a group of SLO definitions. Notice that the unshared instance data are the measurement stream references (`MeasurementStreamInstance`) themselves. It should be stressed that Fresco’s data model accommodates management system elements in addition to the contractual SLA data. As illustrated in Figure 7, the `Profile` which stores shared SLA data (see Figure 4) includes in the `ServiceLevelEvaluationSpecification` `ExtendedData` details on the evaluation configuration formula, configuration directives, and installation details that pertain to the management system. Notice that “we” do not get to define these data, rather they are an artifact of the underlying management system. Notice also that

management system configuration may require the creation of text files that are used as input to management system CLI’s. These data (commands, text files, etc.) must also be represented within the SCOL.

```

<Profile type="tOffering">
  <Name primary="ACMEWebOnDemand" id="4532" />
  <SLAType>External</SLAType>
- <SLOGroupDefinitionGroup>
- <SLOGroupDefinition>
  <Name primary="IP Host+AvailabilityPercent" />
  + <SLOGroup>
- <ServiceLevelEvaluationSpecification>
  <Metric>Availability</Metric>
  <Unit>Percent</Unit>
  <EvaluationInterval>Daily</EvaluationInterval>
  <IntermediateEvaluationInterval>Daily</IntermediateEvaluationInterval:
- <ExtendedData>
  <ExpectedData>false</ExpectedData>
- <EvaluationConfigFormula type="tMetric">
- <metric name="SampleComp_WebsiteAvailability.URL_Website"
  <var>A, state, EP4_URL</var>
  <formula>(* (/ (- (sl_duration) (elapsed_time A s:"Unavailab
    (sl_duration)[1, "Scheduled milliseconds"] ) 100.0 )</formu
  </metric>
  </EvaluationConfigFormula>
- <EvaluationConfigDirective>
- <CustomComponents>
- <Component name="SampleComp_WebsiteAvailability">
- <CustomMetric name="URL_WebsiteAvailability"
  classname="com.company.managed.plugin.typeD.TypeDC
  units="MUnit.MUnit_Nm.PRC" enableTrending="false">
  <Description>Component for URL availability</Description>
  <BreachTypes min="true" max="true" avg="true" total="false
  <BaseMetric name="TimeToAcknowledge" compType="EP4_
  </CustomMetric>
  </Component>
  </CustomComponents>
  </EvaluationConfigDirective>
  <EvaluationConfigLocation>D:/Metrics/WA_Metric</EvaluationConfi
  </ExtendedData>
  </ServiceLevelEvaluationSpecification>
  + <TrendEvaluationSpecification>
  + <MeasurementStreamSpecificationGroup>
  <InternalUseOnly>false</InternalUseOnly>
  </SLOGroupDefinition>
  </SLOGroupDefinitionGroup>
  + <InterSLARelationshipGroup>
  </Profile>

```

Figure 7. SLA Elements Common to a Group of SLAs as represented in Profile.

In concluding, we have found SCOL sufficiently extensible and expressive as we continue to elaborate our SLA management data model (e.g. with business impact and new management system related data) and represents an interesting candidate approach for a broader CMDB.

### 3.2. Fresco use of Form and Style Processors

Fresco exposes input-oriented Forms and output-oriented Styles as shown in Figure 8, leveraging the work associating semantics with XML in [8] to offer a flexible and extensible interface. From the SLA configuration viewpoint the input interaction with the Fresco occurs through form data as a Web Service. Each form is associated with a corresponding form processor which is responsible for realizing the request. The syntactic elements of the form need not associate directly with the underlying system data model, provided the form processor is able to enact the transformation. Conversely, on the output side, the resulting data

elements extracted from the system data model will be rephrased by the style processor into a format readily appreciated by the recipient. E.g. results returned from a given style processor may be in plain text (need not be XML). Fresco’s use of form and style processors exposes to applications a variety of levels of API for various application oriented scenarios.

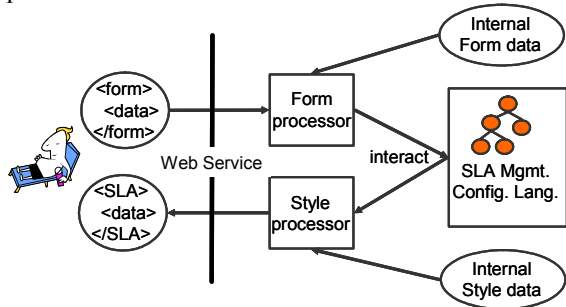


Figure 8. Fresco form/style processor interactions.

### 3.3. Template Driven Client Interface

Users interact with Fresco through a Web Service, as shown in Figure 8 via a portType operation `executeForm` where the “command” (a.k.a. a form, see Section 3.2) is phrased at either a high or low level of abstraction. The result is formatted based on the appropriate style processor and is returned as a response message.

The raw SCOL-based repository update and query interface accepts commands to either “get”, “insert”, “delete” or “update” elements within the model as illustrated by the lower part of a prototype GUI to exercise the Fresco command interface (Figure 9). Since there is rich semantic interpretation for the top level model elements the command interface *does not* support arbitrary XML “document modifications”, rather a controlled vocabulary. For example, the `genericForm` (a “low level” controlled passthru to the SCOL-based repository) in Figure 9, supports commands such as `lang:get lang:at=/Root/ScheduleGroup` which provides back a list of the Schedules along with their IDs. The command shown in Figure 9, `lang:get lang:at=/Root/ScheduleGroup/Schedule[Name[@ID='2121']]` returns the complete Schedule whose ID is 2121 (see Figure 10) with `RelatedToReference` and `IncludeReference` relationships expressed as XPath links (e.g. Figure 11 and discussed later). An alternate command (`getI`) could return the `IncludeReference` links expanded.

We have found that a controlled vocabulary approach to accessing the configurator allows the integrity of the store to be maintained while offering the applications logically relevant commands as per their needs. The upper part of Figure 9 illustrates a “high level” API to the configurator more applicable to SLA management GUIs. Exactly the same data is being requested from the

`simpleForm` as was the case earlier, however, the command syntax is now closely assigned to the “top level” SLA management model elements, e.g. to get a schedule we issue `lang:getSchedule`. It should be appreciated that this new query is transformed by the form processor to the “low level” syntax, which is then emitted to the SCOL-based repository.

Notice also that the schedule is typed (e.g. Figure 10), allowing a variety of different schedule formats to be supported. This particular example is a multi-state (or “non-overlapping”) schedule in which for any particular time interval the “last state” (processing from top to bottom) is used. As will be shown in Section 4, other variations exist which challenge this assumption and further demonstrate the value of Fresco.

#### Simple Query Language:

This is a high-level query language. The associated Form processor in the Configurator will translate it to the low-level XPath-based query language.

```
<form:simpleForm form:version="1.0" style:style="xml" xmlns:form="urn:simpl
  <lang:getSchedule>
    <Name id='2121' />
  </lang:getSchedule>
</form:simpleForm>
```

Submit Simple Query

#### XPath-based Query Language:

This is a standard low-level query language based on XPath. The query is executed unmodified by the associated Form processor in the Configurator.

```
<form:genericForm form:version="1.0" style:style="xml" xmlns:form="urn:gene
  <lang:get lang:at="/Root/ScheduleGroup/Schedule[Name[@id='2121']]"/>
</form:genericForm>
```

Submit XPath-based Query

Figure 9. High-level and low-level APIs to Fresco. Commands are passed in using Web Services requests using extensible forms.

## 4. Application Example

One claim made in Section 2 was that the mapping from the SLA contract data to SLA management system is nontrivial. We illustrate this by reconsidering a scenario discussed earlier, that of mapping overlapping schedules to a leading commercial management system product that supports only non-overlapping schedules. As mentioned, our semantic analysis of many contracts indicates that clauses are sometimes written so as to imply overlapping schedule states. An example of such a schedule is illustrated in Figure 10 (some details collapsed), which is quite similar to the schedule of Figure 4.

As before, the syntax defines a set of states and periods, notice however that the type of the schedule is `OverlappingFirstStateIsUsed`. In other words, the states are to be interpreted as potentially overlapping in time. Again, the schedule is directly represented within SCOL, but as stated earlier, the commercial SLA

management system elements support SLOs in *only one* state at time. Hence, the above contractual clause needs to be mapped into multiple schedules, one for each level of overlap (typically two). Rather than requiring the service personnel be responsible for managing this mapping manually (perhaps using a spreadsheet tool), Fresco is able to *configure* and *maintain track* of the relationships between the multi-state schedule as articulated in the SLA contract and the deployed SLA management schedules. This is illustrated in Figure 11 (state details collapsed) by judicious use of relatedToReferences in the Schedule/InterSchedulingRelationshipGroup element.

```

<Schedule type="OverlappingFirstStateIsUsed">
  <!-- Multi State Schedule -->
  <Name primary="NewStandardAndPeak" id="2121" />
  - <StateGroup>
    <State>CRITICAL</State>
    <State>PEAK</State>
    <State>PRIME</State>
    <State>STANDARD</State>
    <State>LOW_IMPACT</State>
    <State>OFF_HOURS</State>
    <State>NO_SERVICE</State>
  </StateGroup>
  <DefaultState>NO_SERVICE</DefaultState>
  <IsAuxiliary>false</IsAuxiliary>
  <AuxiliaryScheduleGroup />
  + <PeriodGroupGroup>
    - <PeriodGroup state="PEAK">
      - <Period>
        <PeriodStart time="09:00:00" timezone="America/New_York" />
        <PeriodEnd time="16:59:59" timezone="America/New_York" />
        - <Frequency>
          <Weekly days="Monday Tuesday Wednesday Thursday Friday" />
        </Frequency>
      </Period>
    </PeriodGroup>
    - <PeriodGroup state="STANDARD">
      - <Period>
        <PeriodStart time="00:00:00" timezone="America/New_York" />
        <PeriodEnd time="23:59:59" timezone="America/New_York" />
        - <Frequency>
          <Daily />
        </Frequency>
      </Period>
    </PeriodGroup>
  </PeriodGroupGroup>
  + <InterScheduleRelationshipGroup>
</Schedule>

```

Figure 10. SLA data for a multi state schedule.

To the extent that the SLA reporting system can be configured to generate specific reports, Fresco can likewise configure the reports so as to make them closely associate with the original SLA contract.

In fact, the job for Fresco is rather more complex for this example in that the new schedules result in not one but multiple management system offerings (differing in their schedules) and multiple contract instances, as indicated in Figure 12. As with the schedules themselves, these new data elements are stored as new instance data in SCOL and are maintained by Fresco. With Fresco, a change to an element in the SLA contract representation (or its schedule) will be reflected by propagated changes to the automatically generated elements. This level of consistency is hard to guarantee when the mapping is manually done by service personnel.

```

- <Root>
- <ScheduleGroup>
  - <Schedule type="OverlappingFirstStateIsUsed">
    <!-- Multi State Schedule -->
    <Name primary="NewStandardAndPeak" id="2121" />
    + <StateGroup>
      <DefaultState>NO_SERVICE</DefaultState>
      <IsAuxiliary>false</IsAuxiliary>
      <AuxiliaryScheduleGroup />
    + <PeriodGroupGroup>
      - <InterScheduleRelationshipGroup>
        - <InterScheduleRelationship type="MgmtSystemXYZRealization">
          <Schedule relatedToReference="store:/Root/ScheduleGroup/Schedule[Name
            [@primary="NewStandardAndPeak+Autogen00"]]" />
          <Schedule relatedToReference="store:/Root/ScheduleGroup/Schedule[Name
            [@primary="NewStandardAndPeak+Autogen01"]]" />
        </InterScheduleRelationship>
      </InterScheduleRelationshipGroup>
    </Schedule>
  - <Schedule type="NonoverlappingFirstStateIsUsed">
    <!-- Single State Schedule -->
    <Name primary="NewStandardAndPeak+Autogen00" id="2122" />
    + <StateGroup>
      <DefaultState>NO_SERVICE</DefaultState>
      <IsAuxiliary>false</IsAuxiliary>
      <AuxiliaryScheduleGroup />
    + <PeriodGroupGroup>
    </Schedule>
  - <Schedule type="NonoverlappingFirstStateIsUsed">
    <!-- Single State Schedule -->
    <Name primary="NewStandardAndPeak+Autogen01" id="2123" />
    + <StateGroup>
      <DefaultState>NO_SERVICE</DefaultState>
      <IsAuxiliary>false</IsAuxiliary>
      <AuxiliaryScheduleGroup />
    + <PeriodGroupGroup>
    </Schedule>
  </ScheduleGroup>
</Root>

```

Figure 11. Realization of a multi-state schedule using several single-state schedules based on the SLA management system in use.

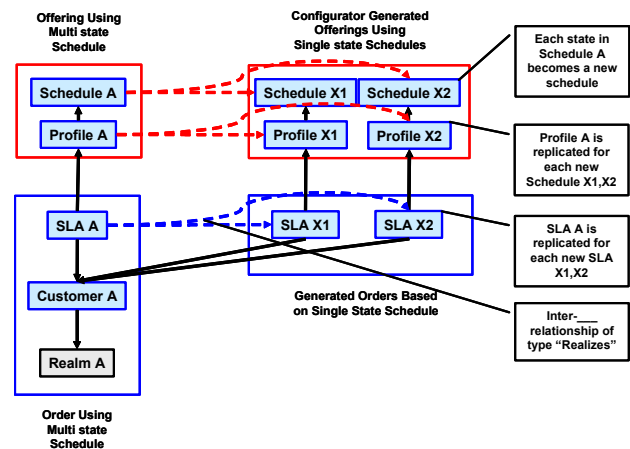


Figure 12. Association between SLA contract and multiple SLA management orders is maintained by the configurator.

To better appreciate the benefits of Fresco to the service personnel, we conclude this section by comparing in Table 1 the number of Web Service requests required to complete representative tasks with and without Fresco. Notice also that whereas command extensions to the underlying system management products is typically outside the users control, command extensions to Fresco is not, and provides a differentiating capability to the service management team.

**Table 1. Required configuration steps, with and without Fresco.**

Task	Management system Web Service commands	Fresco Web Service commands
List data for all SLAs	1 (index) + n (SLAs)	1 (command supported)
List all data related to a particular SLA	1 (customer) + 1 (profile) + 1 (SLA) + n (schedules) + m (realms)	1 (command supported)
List all data associated with a particular overlapping Schedule (as discussed)	1 (overlapping schedule) + n (nonoverlapping schedules) + n (associated profiles) + n (corresponding SLAs)	1 (command supported)

## 5. Related Work

There have been a number of language specifications focused on the representation and processing of SLAs, e.g. SLAng [9], WSLA [10], WSOL [11], and [12,13]. Many are idealized languages expressing contracts as a collection of term and condition clauses, whereas SCOL is a representation of the data elements that have been identified as relevant to SLAs based upon both a review of many contracts (e.g. the abridged SLA in the appendix) and various SLA management systems. Our specification differs in that it is designed to accommodate the mapping between *external agreements* and *internal realizations of the same agreement* as they are deployed in real systems, all in a consistent way.

There have been various SLA management frameworks for e-business including, e.g., SweetDeal [14], DocLog [15], [16] and [17]. These provide multiple contract management features including a contract repository, contract monitoring, etc. The systems provide a set of contract clause templates to facilitate contract provisioning and record signed contract instances and policy documents as XML documents. For instance [18] attempts to automate service level attainment monitoring using a XML-based specification. Often these frameworks omit certain key contractual service level evaluation related details (e.g. quality measure adjudication rules) and assume only a single SLM engine for all attainment result processing. This assumption is challenged in real world situations where the sequence may be handled by different systems.

## 6. Conclusions

While the demand of using Web Services to run and manage software products as services increases, we have found that configuring SLA management software using

Web Services in support of other Web Services applications' SLA compliance monitoring needs is a challenging task. Since agreeable terms on service level specifications cannot be limited in practice, there will necessarily be "gaps" between contractual SLA terms and the required tasks of configuring SLA management software. Configuring such software through Web Services becomes more challenging when considering the need for service offerings to reflect a synthesis of various system products, each of which caters to customer requirements for only one aspect of the complete services requirement. In this paper, we presented the design rationale of the *Fresco* framework and illustrated how it would facilitate configuring extensible SLA management systems using Web Services. Key features of the Fresco framework are:

- it is based on a flexible, extensible XML based SLA management configuration language (SCOL);
- it uses Web Services API to surface the extensibility to the client applications by using forms and style processors, and a template driven client interface;
- it accommodates Web Services and legacy interactions with commercial system management products through product specific adaptors; and
- it uses Web Services API to hide its handling of many configuration commands required to support commercial SLA management contract clauses.

Our experiences with a Fresco-based prototype for a state-of-the-art SLA management product are most promising. They suggest the Fresco-based solution would significantly reduce the complexity of configuring any SLA management software using Web Services. We believe the Fresco framework and SCOL could also be exploited well for other functional areas in an on demand infrastructure and form part of a CMDB.

## References

1. A. Hiles, *The Complete IT Guide to Service Level Agreements - Matching Service Quality to Business Needs*. Rothstein Associates Inc., Brookfield, Connecticut, USA. 1999/2000.
2. N. Frey, R. Matlus and W. Maurer, *A Guide to Successful SLA Development and Management*. Gartner Group, Strategic Analysis Report. 2000.
3. C. Ward, M. Bucu, R. Chang and L. Luan. "A Generic SLA Semantic Model for the Execution Management of e-Business Outsourcing Contracts," *3rd International Conference on e-Commerce (EC-Web 2002)*. 2002.

4. *The IBM Universal Management Infrastructure*, IBM Global Services, IBM Corp. April 2003.
5. C. Molina-Jimenez, S. Shrivastava, J. Crowcroft, and P. Gevros, "On the Monitoring of Contractual Service Level Agreements," *1st International Conference on Electronic Contracting (WEC'04)*. 2004.
6. M. Buco, R. Chang, L. Luan, C. Ward, J. Wolf and P. Yu., *Utility computing SLA management based upon business objectives*. IBM Systems Journal, 43(1). 2004.
7. C. Ward, N. Li, M. Buco, R. Chang, L. Luan, "Integration and Exploitation of SLA Management Data using the DIRECT Metadata Management Framework," *International Conference on Computers, Communications and Control Technologies (CCCT'04)*. 2004.
8. Q. Li, M. Y. Kim, E. So, S. Wood, "XVM: a bridge between xml data and its behavior", *13th International Conference on World Wide Web (WWW'04)*. 2004.
9. D. Lamanna, J. Skene, W. Emmerich, "SLAng: A Language for Defining Service Level Agreements," *9th IEEE Future Trends of Distributed Computing Systems 2003 (FTDCS'03)*. 2003.
10. A. Keller and H. Ludwig, *The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services*, Journal of Network and Systems Management, Special Issue on E-Business Management, 11:1, Plenum Publishing Corp., March 2003.
11. V. Tasic, et al, "The Web Service Offerings Language (WSOL) and the Web Service Offerings Infrastructure (WSOI)," *CASCON 2003 workshop Policy and Service Level Agreements: Their Role in Automated Management in Systems*. 2003.
12. A. Farrell, A. Sergot, M. Salle, C. Bartolini, "Performance Monitoring of Service-Level Agreements for Utility Computing Using the Event Calculus," *First International Conference on Electronic Contracting (WEC'04)*. 2004.
13. V. Kabilan, and P. Johannesson, "Semantic Representation of Contract Knowledge using Multi Tier Ontology," *1st International Workshop on Semantic Web and Databases (SWDB 2003)*. 2000.
14. B. N. Grosz, and T. C. Poon, "SweetDeal: representing agent contracts with exceptions using XML rules, ontologies, and process descriptions," *12th International Conference on World Wide Web (WWW'03)*. 2003.
15. Y. H. Tan and W. Thoen, "DocLog: An Electronic Contract Representation Language," *11th International Workshop on Database and Expert Systems Applications (DEXA)*. 2000.
16. A. Goodchild, C. Herring and Z. Milosevic, "Business Contracts for B2B," *Infrastructures for Dynamic Business-to-Business Service Outsourcing (ISDO 2000)*. 2000.
17. C. Herring and Z. Milosevic, "Implementing B2B Contracts using BizTalk," *34th Hawaii International Conference on System Sciences (HICSS 2001)*, 2001.
18. A. Sahai, et al, "Automated SLA Monitoring for Web Services," *13th IFIP/IEEE International Workshop on Distributed Systems Operations and Management (DSOM 2002)*. 2002.

## Appendix: An Abridged Web Hosting SLA

Provider's Service Level Agreement (SLA) standard for Customer's Web Hosting environment is less than four hours per calendar month of downtime, which is an availability of approximately 99.5 percent. This SLA objective applies to downtime caused by Provider in regard to operation of system software, loading of system software, hardware failure, backup and recovery of files, and connectivity from the server farm to the Internet and from the server farm to the Customer data center. This SLA objective specifically does not include failures caused by Customer, outages associated with contract maintenance provisions, failure of bandwidth connectivity, and external failures outside the Web-site Hosting Environment.

Availability for the purposes of the SLA objective is based either on Help Desk trouble-ticket information or Provider-detected downtime. For problems reported to the Provider Help Desk trouble-ticket system, opening of the Help Desk trouble ticket establishes the outage begin time; the problem resolved time, as documented in the Help Desk trouble ticket, establishes the end time for that particular outage. For Provider-detected downtime, the outage begin time is based on the first detection of any outage, and the end time for the outage is based on the problem resolve time. The amount of downtime in each calendar month will be totaled to determine any failure to meet the SLA objective. Table 1 documents the monthly refunds or premiums associated with missing or exceeding the SLA standard for each calendar month. In no case shall more than the monthly charge be credited for downtime incurred in a single month.

Credits and premiums will be aggregated and settled on an annual basis. Premiums may be used to offset credits, but will not create an obligation on the part of a customer to pay more than those monthly charges documented in this statement of work (SOW) or subsequent transaction document pertaining to these services. Any payment due from Provider to Customer will be paid by January 31st of the following year.