Concept of MeML, MESP, and WME Service

1. WME Environment
2. MeML
   2.1 MeML Basics
   2.2 MeML Elements
      2.2.1 Computation Elements
      2.2.2 Control Elements
      2.2.3 Education Elements
      2.2.4 Content Elements
      2.2.5 Organization Elements
3. MESP
   3.1 MESP Basics
   3.2 Service Mode
   3.3 MESP Event
   3.4 MESP Message and MESP Packet
   3.5 Internal Layer
   3.6 Protocol Interface

1. WME Environment

Education content is created with ‘MeML+HTML+MathML+SVG’. MeML page requires the support of MeML Processor to generate lively page to final users. MeML specification defines the functions and usage of every MeML tag. The specification doesn’t care how MeML Processor implements these functions. And we are not emphasizing whether MeML Processor is at client-side in Internet browser or at server-side behind web server. Many important functions of MeML require the support of backend WME Service. WME Service could be supplied in many ways using existing or emerging server-side technology. No matter what kind of technology is used to create WME service, each service must have a WME Description File to indicate its service detail. MeML Processor is responsible for understanding and handling each kind of WME service. MESP protocol is designed for the communication between MeML Processor and WME Service or between
WME Services. As the requirement of WME Framework and language independence of MeML, MeML Processor is not forced to use MESP as the only communicating protocol. But MESP is designed to provide much more powerful, convenient, and direct support to MeML language processing. MESP can also be used in other circumstance other than WME Framework.

2. MeML

2.1 MeML Basics

MeML is designed to help educator creating/publishing online courseware in the form of XML pages on Internet. Taking the advantages of XML language, courseware written in MeML is easy to save, transfer, search, reuse, and understand because it is in plain text. No binary code means no binary-compliant problem. Such XML-based courseware can be downloaded to anywhere and executed in any popular Internet browser. And it is easy to be created by teachers with basic knowledge of web pages.

MeML is more than just a descriptive XML language. MeML can not only describe the structure/organization of education materials, but also support in-page math computation, invoke Internet educating service (WME Service), control the access of MeML page content with the distinction of user types (student or teacher), and regulate the component of MeML page for easy customization and reuse.

MeML features list:
1. Structuring and organizing education content - text, demo, reference, example, etc.
2. Supporting in-page mathematics computation.
3. Global configuration and import/export to simplify the context setup of MeML page.
4. Supporting distinction between student user and instructor user
5. Supporting multiple different views of same MeML page with regarding to the type of user.
6. Supporting customization of page component when import/export part or whole page to a different context.

With support of several general WME services, MeML also support:
1. User authentication
2. Assessment /grade
3. Editable page
4. Controlled access

2.2 MeML Elements

2.2.1 Computation Elements
MeML Computation Elements include 11 tags: <variable>, <expression>, <assign>, <condition>, <with>, <true>, <false>, <computation>, <range>, <function>, and <mathgraph>. With these elements, author can perform flexible and dynamic in-
page math computation, control the computation using simple loop and conditional jump, and display the computing result in MathML.

**<variable>**
Define a variable for in-page math computation.

*Attributes*: id, type  
*Events*: onModified

Example:

```xml
<variable id="cns" type="integer" />  
<variable id="gs" type="real" onModified="step1">2.03</variable>
```

The first example define a variable with name of “cns” and type “integer”. Its initial value is 0 by default. “cns” is also the page ID of current tag <variable> in MeML page. The second example is similar to the first one. The difference is that it is given an initial value “2.03”. Any change to the value of “gs” will trigger a computation which can be referenced by page ID “steps”.

**<expression>**
Define an expression for in-page math computation. The expression can be written in different encodings like MathML, infix, OpenMath or in the language format of some popular mathematics software like Maple, Maxima, Mathematica, etc.

*Attributes*: id, encoding  
*Events*: onModified

Example,

```xml
<expression id="cnsamt" encoding="mathml">
  <math>
    <apply>
      <times/>
      <ci>cns</ci>
      <cn>1.95</cn>
    </apply>
  </math>
</expression>

<expression id="total" encoding="infix">
  cnsamt+gsamt+csamt+abbamt
</expression>
```

The name of variable can be used directly in expression no matter what kind of encodings the expression is.

The definition of expression doesn't mean a computation on the R-value of the expression will happen immediately. When to compute the expression is decided by if the “<expression>” is put or used in a “<computation>” block.
<assign>
Assign a value to a computation element. The value is given between “<assign>” and “</assign>” tags. The type of value should be same as the type of object pointed by attribute “rid”.

*Attributes*: rid

*Example*,

   <assign rid="gsamt">gs*2.96</assign>
   <assign rid="total"> cnsamt+abbamt </assign>

</assign>

<condition>
Define a condition which can be used to decide next computation step. It requires at least two MeML computation elements as its children.

*Attribute*: id, judge

*Example*,

   <condition id="cond1" judge="<" >
   <use rid="gs" />
   <use rid="constant" />
   </condition>

The example defines a condition which is be TRUE if variable “gs” is larger than variable “cns”, otherwise it is FALSE.

The test of condition doesn’t begin when the condition is defined. The test is invoked when a <with> element is executed in a computation blocks.

<with>
Define a “if .. then .. else” structure to control the next step of computation with regarding to result of a condition.

*Attributes*: id, condition

*Events* onTrue, onFalse

*Example*,

   <with id="judge1" condition="cond1">
   <true> <use rid="step1" /> </true>
   <false> <use rid="step2" /> </false>
   </with>

   <with id=" judge2" condition="cond1" onTrue="step1" onFalse="step2" />

Above two examples actually do the same thing, but the first example can have wider usage than the second. Events "onTrue" and "onFalse" will arouse execution of some computation blocks.

<computation>
Define a in-page computation block which may require a remote WME Computation Service to complete.

Attributes: id, wmeurl
Events: onExecute

Example,
<computation id="step1" wmeurl="http://wme.cs.kent.edu/maple.wme">
    <assign rid="gs">gs+1</assign>
    <assign rid="gsamt">gs*2.96</assign>
    <use rid="total" />
    <with id="judge3" condition="cond1" onTrue="step1" />
</computation>

When the above computation is triggered for some reason, it will assign a new value to variable “gs” and expression “gsamt”. Then it will update the value of expression “total”, because “total” depends on “gsamt” – “total = cnsamt+gsamt+csamt+abbamt”.

Last, it checks the condition “cond1”. If condition is TRUE, the whole computation block is executed again, otherwise, the computation ends up here.

<true>
Define a TRUE block which must be a child element of <with>

<false>
Define a FALSE block which must be a child element of <with>

<range>
Define a range in real domain.

Attributes: id, from, to, type
Events: onModified

Example,
<range id="r1" from="0" to="10" type="open" />
<range id="r2" from="cns" to="gs" type="close"/>

The first example defines a range (0, 10) using constant. The second defines a range using the value of two variables/expressions. If cns=2, gs=4, then the range is [2, 4].

<function>
Define a function with given variables and expression

Attributes: id, alias, paramnum
Events: onModified

Example,
<function id="f" alias="sin" paramnum="1">
The first example defines function \( f(x) = \sin(x) \). The second defines function

\[
g(x, z) = f(x) + z = \sin(x) + z
\]

The first example defines function \( f(x) = \sin(x) \). The second defines function \( g(x, z) = f(x) + z = \sin(x) + z \).

**<mathgraph>**
Generate graph with given function(s). It may require a remote WME Computation Service to create graph data.

Attributes: id, wmeurl, width, height, operation, function
Events: onModified

Example,

\[
<\text{mathgraph id="plot" wmeurl="http://wme.cs.kent.edu/maple.wme" width="400" height="300" operation="plot2d" function="f"} />
\]

### 2.2.2 Control Elements

There are 14 MeML control elements – \(<\text{use}>\), \(<\text{userinput}>\), \(<\text{parameter}>\), \(<\text{configure}>\), \(<\text{wmeservice}>\), \(<\text{session}>\), \(<\text{certifieduser}>\), \(<\text{import}>\), \(<\text{editable}>\), \(<\text{sensitive}>\), \(<\text{controlled}>\), \(<\text{topicobject}>\), \(<\text{part}>\), and \(<\text{manipulate}>\). They give powerful system support as well as maximum flexibility of code reusing, page customization to MeML page author.

\(<\text{use}>\)
Introduce the usage of another MeML code block, whose root element’s ‘id’ equals ‘rid’, to current position. The type of usage is defined by the ‘type’ attribute that has 3 possible values – ‘reference’, ‘display’, and ‘copy’. The default is ‘reference’. If the MeML code block belongs to a Topic Object, then the ‘topic’ attribute should be used and the usage type must be ‘copy’.

Attributes: rid, type, topic

Examples,

```
<use rid="gs" />
```
Reference the value of variable ‘gs’ in current position. This does not result in any change of MeML page content.

```
<use rid="total" type="display" />
```
Display the expression with ‘id’ equals ‘total’ in current position. This results in the change of page content. But it will not create another expression with same ‘id’. (Two <expression> with same ‘id’ in one page violates the XML standard). Author can look at this kind of usage as a place-holder for display purpose. Once original expression is changed, the displays of all place-holders are changed either.

```
<use rid="blockid" type="copy" topic="pizza-grid" />
```
Copy a block of MeML code from a Topic Object – “pizza-grid” to current position. The root element of source code uses ‘blockid’ as value of ‘id’ attribute. This results in the change of page content

```
-userinput-
```
Create an input tool for user of current MeML page. The input received from user is used to update the value of a MeML element like ‘variable’, ‘expression’. The MeML element’s ‘id’ is same as ‘name’ attribute of <userinput>. The look-and-feel of page object created by <userinput> is various according to the ‘type’ and ‘style’ attribute of <userinput>.

Attributes: name, type, style
Events: onModified

Example,

```
-userinput type="expression" name="exp1" />
```
Get input and use the input to change definition of expression ‘exp1’.

```
-parameter-
```
Define an attribute of direct parent as a parameter. When the parameter is changed, the value of direct parent’s attribute is changed too.

Attributes: name

Example,

```
<mathgraph id="plot" wmeurl="http://wme.cs.kent.edu/maple.wme "
```
Declare global configurations which influence whole MeML page. All configurations are included in `<configure>` and `</configure>` tags. Configurations include: “wmeservice”, “session”, “certifieduser”, etc. (Note, “Math Player” could also be considered as a configuration. But we suppose MeML Processor should have the ability to detect this configuration when it is installed on local machine.)

Attributes: id

<wmeservice>
A global configuration defines a WME Service which could be accessed by any MeML code in current page. MeML code can override this configuration by giving its own preferred service provider through attribute ‘wmeurl’. Typical WME Services include “Math Engine”, “Authenticate”, “Broadcast”, “History Record”, “Directory”, and “Terminology”. The detail of WME Services is discussed in section 4.

Attributes: id, type, wmeurl

Example,

```xml
<wmeservice id="maple" type="Math Engine" wmeurl="telnet://b4.cs.kent.edu:40000/" />
```

<session>
Declare session for local user when he visits a group of MeML pages. The ‘session’ concept in MeML is different with ‘session’ concept in PHP because the first may defined and controlled at client side, not server side. If that is true, client always know what he is doing. Client doesn’t need to distinguish himself with other users when visiting same MeML page. So, a session is meaningful only when it is connected with a certified user and keeps some information alive when user is browsing through a collection of MeML pages.

Attributes: id, connect, persistence, time

Example,

```xml
<certifieduser id="cert1" require="instructor" encrypt="true" />
=session id="s1" connect="cert1" persistence="true" time="0:05" />
```

With ‘persistence’ set to ‘true’, the user’s certificate gotten from ‘Authenticate’ service can be save in local machine and be effective across other MeML page visited latter. Then user need not supply authentication info for each MeML page he visits. Session could also have time limit. Above example set time limit to be 5 minutes before the visit of next MeML page.
<certifieduser>

Ask visitor of current MeML page to provide security information (user name, password) and verify that information through “Authenticate” service. The service location may be given by either '<wmeservice>' configuration or “wmeurl” attribute of '<certifieduser>' tag itself.

Attributes: id, wmeurl, utype, uname, require, link, encrypt,

Events: onSuccess, onFail

Example,

<wmeservice id="auth" type="Authenticate"
    wmeurl="http://kimpton.kent.edu/auth.wme"/>
    
    <certifieduser id="cert1" utype="instructor" encrypt="true" />
    
    <certifieduser id="cert2" utype="student" encrypt="true" require="cert1" />

This example asks WME Authenticate Service to verify that current user type is “instructor”. The popup dialog may appear to ask for user name and password that are required by “Authenticate” service. The information transmitted during authentication must be encrypted. If verification is successful, a certificate returned from “Authenticate” service should be saved by MeML Processor for future use. More than one '<certifieduser>' can appear in the page to allow more users to visit this page. If all '<certifieduser>' fail to return an effective certificate, the access to current page should be forbidden. Attribute ‘require’ points to another '<certifieduser>' tag. It adds a condition to verification process. It means ‘Authenticate’ service should have issued another alive certificate to someone else. That certificate meets the requirement introduced by ‘require’ attribute. And that certificate must be alive by now. (not unregistered).

A certificate should include following information: 'id' of current '<certifieduser>', user type, user name, location of authentication service, generated secret code to identify the user, time limit of certificate.

If there is already one alive certificate existing, '<certifieduser>' will do nothing and accept the certificate as current user identification.

A blank certificate may be created by MeML Processor to perform some universal operations only if only there is no '<certifieduser>' defined in MeML page.

<import>

Import a MeML code block from another MeML page. It could be the whole page or a part of the page. Usually, the imported MeML code will not be inserted into current position immediately. It can be inserted into anywhere of current page as long as a '<use>' tag references the content in imported MeML code. Typically, '<import>' tag is used when author wants to reuse Topic Objects declared in another MeML page, or to use same configuration context of that MeML page.

Attributes: rid, pageurl, content

Example,

<import rid="step1", pageurl="http://wme.cs.kent.edu/algebra.meml"
Import a MeML code block from MeML page pointed by 'pageurl'. The root element of code block should have “step1” as “id” attribute. The content of code block is introduced into current page as “text” without any check.

<editable>
Define a block of XML code to be editable by the user of the page. If user edits the content, the new content can be saved for future use only when WME Service – “History Record” is available. If no required WME Service available, the content should not be displayed as editable. MeML Processor may or may not support the content encoding contained in ‘<editable>’ and ‘</editable>’ tags;

Attributes: id, controlledby, availableto
Events: onModified

Example,
<configure>
  <wmeservice id="auth" type="Authenticate"
               wmeurl="http://kimpton.kent.edu/auth.wme" />
  <wmeservice id="database" type="History Record"
               wmeurl="http://kimpton.kent.edu/history.wme" />
  <certifieduser id="cert1" require="instructor" encrypt="true" />
  <certifieduser id="cert2" require="student" encrypt="true" />
</configure>

<editable id="edit" controlledby="cert1" availableto="cert2">
  <p> Editable Content </p>
</editable>

The example defines a block of content that can be edited by ‘instructor’ and can be read by either ‘instructor’ or ‘student’. When MeML Processor finds ‘<editable>’ tag, it should check if there is an available certificate associated with current user. If user is instructor, MeML Processor should show the editable content in an editing window. When instructor submits the edited content, the new content as well as the certificate, the URL of current page, ‘id’ of ‘<editable>’ tag, and tag itself should be sent to WME History Record Service.

If the certificate shows current user is student, MeML Processor should check the availability of any new update of the content by sending the certificate, the URL of current page, ‘id’ of ‘<editable>’ tag, and tag itself to WME History Record Service. If there is no any update available, then the original content in MeML page should be displayed. If there is an update, then the update is displayed.

How to verify/accept the certificate is up to the providers of WME History Record Service and WME Authenticate Service.

<sensitive>
Define a MeML code block which is only visible to user with specified certificate. The function of sensitive requires WME Authenticate Service. If no required WME Service available, the content should not be displayed.
Attributes: id, availableto

Example,

```xml
<certifieduser id="cert1" require="instructor" encrypt="true"/>

<sensitive availableto="cert1">
  <p> Instructor notes </p>
</sensitive>
```

**<controlled>**

Define a MeML code block which access can be controlled – “show/hide” - by specified user. To function of `<controlled>` element requires WME Broadcast Service and WME Authenticate Service. If no required WME Service available, the content should not be displayed.

Attributes: id, controlledby, visible

Example,

```xml
<wmeservice id="auth" type="Authenticate"
  wmeurl="http://kimpton.kent.edu/auth.wme"/>
<wmeservice id="broad" type="Broadcast"
  wmeurl="http://kimpton.kent.edu/broadcast.wme"/>
<certifieduser id="cert1" utype="instructor" uname="kim" encrypt="true"/>
<certifieduser id="cert2" utype="student" encrypt="true" link="cert1"/>

<controlled id="control1" controlledby="cert1" intial="hide">
  <question> Question requires controlled access </question>
</controlled>
```

In this example, MeML Processor should check the certificate of current user and register returned certificate and IP address to WME Broadcast Service. ‘link’ attribute declares a link relation to another possibly alive certificate at any WME Service which requires certificate.

If user is student, Processor continues to check the value of attribute “initial”. If it is “hide”, Processor put a place-holder at current position and waits for further information – an activation event on the controlled content.

If user type is “instructor”, MeML Processor should display the controlled content as well as a control (could be a button) beside the content. If instructor enables the content using given control, an activation event is fired to WME Broadcast Service. The event should contains event type, event data, user’s certificate, the URL of current page, the ‘id’ of MeML tag that fires the event, and the tag itself. The service should check if there is anyone who cares about this event through certificates’ link relation. Then it broadcasts the event to all listeners. When MeML Processor of student user receives activation event, it checks if user still stays on the same page by checking URL, tag, and ‘id’. If user is waiting for the controlled content, it shows the content; otherwise, it buffers the event for future use.
Define a Topic Object that can be used in current page or be exported to other MeML pages.

A Topic Object is a standalone MeML code block that can complete some functions all by itself. Author can put anything that support the function of Topic Object in it – MeML, MathML, HTML, JavaScript, Java Applet, Style Sheet.

The definition of Topic Object itself will not add anything to the MeML page. The only way to use Topic Object is through '<use>' tag. Topic Object can be subdivided into several parts in any way that obeys XML specification. A more general and very flexible usage is using '<use>' tag to distribute each part to a different position of MeML page.

The content of Topic Object can also be created using '<import>' tag to grab page content from other MeML pages and non-MeML pages. The imported content could be anything that can be located through page URL plus ‘id’/’name’ attribute. So, ‘Topic Object’ + ‘Import’ is a very powerful tool to reuse code in non-MeML pages.

Attributes: id, divisible, keyword

Example,

This is an abstract example from WME Pilot Project for Kimpton Middle School. The topic is teaching fraction concept using pizza.

```xml
<topicobject id="pizza" divisible="true" keyword="fraction">
  <part id="p1" importance="required" duplicable="false">
    <!-- JavaScript code that does calculation and controls the display of pizza -->
  </part>

  <part id="p2" importance="recommend" duplicable="false">
    <!-- HTML/JavaScript code that receives user input values for the dimension of the pizza -->
  </part>

  <part id="p3" importance="required" duplicable="false">
    <!-- HTML code that displays the pizza -->
  </part>

  <part id="p4" importance="recommend" duplicable="false">
    <!-- HTML code that displays the calculated fraction -->
  </part>
</topicobject>
```

Author can utilize '<use>' tag to distribute each part of Topic Object ‘pizza’ to different location in page with maximum flexibility. For example,

```xml
<use rid="p3" type="copy" topic="pizza" />
```

will copy any content included in <part id="p3"> to current position. Because Topic Object is standalone and behave all by itself, the distribution will not break its
application logic. Because the content is copied to destination page, the configuration context of destination page should also apply to the copy.

**<part>**

Define a part in a Topic Object, which could be put anywhere in MeML page. ‘importance’ equals to ‘required’ means if the Topic Object is used in a page, then this part must be included to make sure the Topic Object works. ‘duplicable’ equals to ‘true’ means this part can have multiple copies in same page without generating error.

Attributes: id, importance, duplicable

Example is same as ‘<topicobject>’.

**<manipulate>**

This tag must be used as direct child of ‘<use>’ and ‘<import>’ elements. Its function is changing the definition of a XML code block or attribute of a XML tag, which belongs to the MeML code generated by ‘<use>’ or ‘<import>’ tags. One of its typical usage is to customize the Topic Object imported from another MeML page.

Attributes: rid

Example,

```
<use rid="p4" type="copy" topic="pizza">
  <manipulate rid="someid">
    <!-- New content -->
  </manipulate>
  <manipulate rid="someid">
    <parameter name="param">New Value</parameter>
  </manipulate>
</use>
```

This example changes the definition of part 4 of Topic Object – “pizza” when it is used in current page. It also shows how to change an attribute value of an arbitrary XML element.

**2.2.3 Education Elements**

Currently, there are 7 MeML Education Elements. Unfortunately, not all education elements have very clear definition.

**<exercise>**

Setup an exercise for students

Attributes: id, type, title, keyword

Example is same as the one of ‘<question>’ element.
<question>
Define question and/or its answer based on the type of question. If author wants to save the students’ answer for ‘assessment’ and ‘diagnosis’ in the future, he should declare WME Authenticate Service and WME History Record Service in configuration.

Attributes: id, type, answer, solution

Example,
<exercise id="training" title="Discussion" keyword="fraction">
  <question id="q1" type="verify" answer="a1" solution="labor">
    <variable id="a1" type="real"/>
    <expression id="labor">total*0.5</expression>
    If 50% of the cost of your meal actually goes to pay labor at the restaurant, what is the labor cost for your meal?
    <userinput name="a1" type="real" style="check"/>
  </question>
</exercise>
</test>

Setup context of test for students.

Attributes: id, title, keyword, timelimit
Events: onInterrupt, onComplete

<assessment>
This element is designed for instructor and it should be put under ‘<sensitive>’ element. Its function requires WME Authenticate Service and WME History Record Service to retrieve the answers of students to questions in a specified MeML page.

Attributes: id, roster, pageurl, target

Example,
<assessment id="assess" roster="roster1" pageurl="URL" target="training"/>

‘<assessment>’ depends on ‘<roster>’ to generate a list of students whose work need to be assessed. The work could an ‘exercise’ or ‘test’ that can be located by ‘pageurl’ and ‘target’. The result of assessment could be saved by WME History Record Service.

<diagnosis>
Create a report about student’s performance using the result of assessment. Its function requires WME Authenticate Service and WME History Record Service.

<example>
Generate lively example with given condition. (Note, the function of this element is still unclear and requires more investigation)
Prepare a remediation for student with regarding to the result of '<diagnosis>'.

2.2.4 Content Elements
Content elements help author build up all kinds of knowledge in MeML page from simple to advance. Now, there are 6 Content Elements defined.

<concept>
Create a definition for a concept

Attributes: id, name, keyword

<skill>
Give a description to a skill requirement to students.

Attributes: id, name, keyword

<terminology>
Give definition to a term. The definition could also be generated by WME Terminology Service.

Attributes: id, name, wmeurl, keyword

<equation>
Define an equation.

Attributes: id, name, keyword

<diagram>
Create a diagram.

Attributes: id, caption, keyword

<theorem>
Define a theorem.

Attributes: id, name, keyword

2.2.5 Organization Elements
Organization elements help author create well-organized, ready-for-search MeML page. There are 7 Organization Elements defined.

<abstract>
Define a piece of page content as abstract.

Attributes: id, title, keyword
<lesson>
Declare a lesson.

Attributes: id, title, keyword

<syllabus>
Define syllabus for lesson

Attributes: id, lesson

<roster>
Prepare a student list for a lesson

Attributes: id, lesson

<guide>
Define a guide to lesson

Attributes: id, lesson

<hint>
Prepare a hint for a content element or education element

Attributes: id, for

<summary>
Define a summary for lesson.

Attributes: id, lesson

3. MESP

3.1 MESP Basics

Mathematics Education Service Protocol (MESP) is designed to support communication between MeML Processor and WME Services or between WME Services only. Considering the language characteristics of MeML, it has a kind of Document Object Model and it depends on various events to cooperate on education tasks. The events could be fired to in-page objects or to remote WME Service. So, MESP is designed to focus on event-handling in distributed environment and it is very natural to map MeML Event to MESP Event.

For above reason, MESP looks like a kind of distributed object computation protocol such as SOAP, RMI, CORBA. But there is great difference between MESP and other protocols.
First, the communication carrier of MESP is Event, not remote method invocation. This is biggest difference between MESP and other protocol. One event could be monitored and accepted by multiple receiver and may result in several different service responses in parallel. Remote method invocation can only be sent to one object instance and complete one method call. So, event is much more suitable to schedule backend service automatically.

Second, the message body of MESP is encoding in XML.

Third, MESP support two different working modes (symmetric, asymmetric) to accommodate as more service types as it can to protect server-side investment.

3.2 Service Mode

Figure 1: MESP Symmetric Mode
3.3 MESP Event
MESP Event is defined according to all possible events among MeML Processor and WME Services. Here is a list of MESP Events defined so far at current developing stage. More events may be added in the future.

Compute Event
Request WME Computation Service.

Authenticate Event
Require WME Authenticate Service

Register Event
Register event source as a potential participant of a WME Service

Activation Event
Fire a signal that someone may care

Record Event
Publish a piece of information onto network which might be retrieved in the future.

Search Event
Ask for a piece of information with certain conditions
Each event will result in a returned event by default if the event is accepted by somebody on network. The returned event is usually used for acknowledgement purpose only. Every communicating entity using MESP should provide an interested event list to its local MESP Driver to catch the information it wants. That means a client may reject a service response and pick up others. This may result in inefficiency of communication but it also provides more flexibility to MESP user and help them simplify application logic.

3.4 MESP Message and MESP Packet
MESP Message is simply an encapsulation of MESP Event. The information carried by MESP Event includes:
- Event Type
- Event Name
- Event Data
- Event Time (Logic Time to keep Event Order)
- Event Source (Page URL and Tag ID for MeML Processor)
- User Certificate

The format of Event Data is defined by Event. Here is a sample of MESP message.

```xml
<event type="etype" name="ename">
  <data scheme="url">
    <record id="r1"> … </record>
  </data>
  <time tick="logic time" />
  <scoure pageurl="url" tag="tagname" id="tagid" />
  <certificate>
    <!-- data belongs to certificate -->
  </certificate>
</event>
```

MESP Packet contains both MESP Message and the data for mapping service call. The data format of mapping service call could be various according to the mapping rules defined in WME Service Description.

3.5 Internal Layers
MESP has two internal layers as shown in figure 3. The “Service Descript Layer” is responsible for understanding the detail of WME Service and mapping MESP event to one or more service call to remote WME Services.

The “Group Communication Layer” is responsible for actual communication between MESP entities. MESP user could also call its function directly to apply more control to underplayed communication.

“CEM” is a file which contains all objects and events definition used by MESP. At current developing stage, “CEM” is just an idea. It may get more usage in the future. It is supposed that all objects and events referenced in WME Description File should come from “CEM”.

3.6 Protocol Interface

Figure 3: Internal Layer of MESP

Three Working Mode:
1. Direct
2. Event Listener
3. Event Listener w. Scenario

Figure 4: MESP Driver

Figure 4 shows the Interface between MESP user and driver. The only thing user application need do is inherit and implement ‘Event Listener’ provided by MESP Program API and then fire events that it wants to put on network.