WSP - Wireless Session Protocol

- **Goals**
  - HTTP 1.1 functionality
  - Request/reply, content type negotiation, ...
  - Support of client/server, transactions, push technology
  - Key management, authentication, Internet security services
  - Session management (interruption, resume,...)

WSP protocols

- **Connection mode** (uses WTP)
  - Session Management (class 0, 2)
  - Method Invocation (class 2)
  - Error Report
  - Push (class 0)
  -Confirmed Push (class 1)
  - Session suspend/resume (class 0, 2)

- **Connectionless mode** (uses WDP or WTLS)
  - Method Invocation
  - Push
    (in general unreliable)

WSP/B session establishment

- S-Connect.req
  (SA, CA, CH, RC)
- Connect PDU
- S-Connect.ind
  (SA, CA, CH, RC)
- S-Connect.cnf
  (SH, NC)
- S-Connect.conf
  (SH, NC)
- WTP Class 2 Transaction
WSP/B session suspend/resume

- **S-Suspend.req** (client S-SAP) to **S-Suspend.ind** (server S-SAP)
- **S-Suspend.ind (R)**
- **Reply PDU**
- **S-Resume.req (SA, CA)** to **S-Resume.ind (SA, CA)**
- **Resume PDU**
- **S-Resume.cnf**
- **WTP Class 2 transaction**

WSP/B session termination

- **S-Disconnect.req (R)**
- **S-Disconnect.ind (R)**
- **WTP Class 0 transaction**

WSP/B method invoke

- **S-MethodInvoke.req (CTID, M, RU)**
- **Method PDU**
- **S-MethodInvoke.ind (STID, M, RU)**
- **Reply PDU**
- **S-MethodInvoke.cnf (CTID)**
- **S-MethodResult.req (STID, S, RH, RB)**
- **S-MethodResult.ind (STID, S, RH, RB)**
- **S-MethodResult.res (CTID)**
- **S-MethodResult.cnf (CTID)**
- **WTP Class 2 transaction**
WSP/B over WTP - method invocation

WSP/B over WTP - asynchronous, unordered requests

WSP/B - confirmed/non-confirmed push
WSP/B over WDP

client

S-Unit-MethodInvoke.req
(SA, CA, TID, M, RU)

server

S-Unit-MethodInvoke.ind
(SA, CA, TID, M, RU)

S-Unit-MethodResult.ind
(CA, SA, TID, S, RH, RB)

S-Unit-Push.ind
(CA, SA, PID, PH, PB)

S-Unit-MethodInvoke.ind
(SA, CA, TID, M, RU)

S-Unit-MethodResult.req
(CA, SA, TID, S, RH, RB)

S-Unit-Push.req
(CA, SA, PID, PH, PB)

Method PDU

Reply PDU

Push PDU

WDP UnitData

service

WAE - Wireless Application Environment

- Goals
  - network independent application environment for low-bandwidth, wireless devices
  - integrated Internet/WWW programming model with high interoperability

- Requirements
  - device and network independent, international support
  - manufacturers can determine look-and-feel, user interface
  - considerations of slow links, limited memory, low computing power, small display, simple user interface (compared to desktop computers)

- Components
  - architecture: application model, browser, gateway, server
  - WML, XML-Syntax, ...
  - WMLScript: procedural, loops, conditions, ... (similar to JavaScript)
  - WTA: telephone services, such as call control, text messages, phone book, ... (accessible from WML/WMLScript)
  - content formats: vCard, vCalendar, Wireless Bitmap, WML, ...

WAE logical model
Wireless Markup Language (WML)

- WML follows deck and card metaphor
  - WML document consists of many cards, cards are grouped to decks
  - A deck is similar to an HTML page, unit of content transmission
  - WML describes only intent of interaction in an abstract manner
  - Presentation depends on device capabilities

- Features
  - Text and images
  - User interaction
  - Navigation
  - Context management

WML Structure

```xml
<WML>
  <CARD NAME="eCard">
    <DO TYPE="ACCEPT">
      <GO URL="/submit?N=$(N)&S=$(S)"/>
    </DO>
    Enter name: <INPUT KEY="N"/>
    Choose speed: <SELECT KEY="S">
      <OPTION VALUE="0">Fast</OPTION>
      <OPTION VALUE="1">Slow</OPTION>
    </SELECT>
  </CARD>
</WML>
```

Deck of Cards

```xml
<WML>
  <CARD NAME="eCard">
    <DO TYPE="ACCEPT" LABEL="Next">
      <GO URL="#card2"/>
    </DO>
    Acme Inc. Directory
  </CARD>
  <CARD NAME="card2">
    <DO TYPE="ACCEPT">
      <GO URL="?send=$type"/>
    </DO>
    Services
    <SELECT KEY="type">
      <OPTION VALUE="em">Email</OPTION>
      <OPTION VALUE="ph">Phone</OPTION>
      <OPTION VALUE="fx">Fax</OPTION>
    </SELECT>
  </CARD>
</WML>
```
The DO Element
- Binds a task to a user action
  - Action type: ACCEPT, OPTIONS, HELP, PREV, DELETE, RESET
  - Label: Text string or image (optional)
  - Task: GO, PREV, REFRESH, NOOP
  - Destination: URL

```xml
<DO TYPE="ACCEPT" LABEL="Next">
  <GO URL="http://www.mysite.com/myapp.wml"/>
</DO>
```

Anchored Links
- Bind a task to the ACCEPT action, when cursor points to a link
  - TITLE= sets the label string (default = "Link")
  - Links are not allowed in select list options

```xml
<CARD>
  Please visit our
  <A TITLE="Visit">
    <GO URL="home.wml"/>home page</A>
  for details.
</CARD>
```

The TEMPLATE Element
- Defines actions & events for all cards in a deck

```xml
<WML>
  <TEMPLATE>
    <DO TYPE="OPTIONS" LABEL="Main">
      <GO URL="main_menu.wml"/>
    </DO>
  </TEMPLATE>
  <CARD NAME="msg1">
    <DO TYPE="ACCEPT" LABEL="Next">
      <GO URL="#msg2"/>
    </DO>
    First story
  </CARD>
  <CARD NAME="msg2">
    Second story
  </CARD>
</WML>
```
The SELECT Element

- Display a list of options
  - Each option may set the KEY variable and/or bind a task to the ACCEPT key
  - TITLE= dynamically sets the label string
  - MULTIPLE="TRUE": Allows user to pick multiple items

```xml
<SELECT KEY="city">
  <OPTION VALUE="ber">Berlin</OPTION>
  <OPTION VALUE="rom">Rome</OPTION>
  <OPTION TITLE="Find" ONCLICK="find.cgi">New City</OPTION>
</SELECT>
```

The INPUT Element

- Prompts user to enter a string of text
  - DEFAULT=key_value: Default KEY variable (displayed to user)
  - FORMAT=format_specifier; If omitted, free-form entry is allowed
  - EMPTYOK="TRUE": Browser will accept null input
  - TYPE="PASSWORD": Special entry mode handled by the browser
  - MAXLENGTH=number; Maximum number of allowed characters

```xml
<INPUT KEY="ssn" FORMAT="NNN\-NN\-NNNN"/>
```

WML – example I

```xml
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
 "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="card_one" title="simple example">
    <do type="accept">
      <go href="#card_two"/>
    </do>
  </card>
</wml>
```

Forecast
1  Berlin
2  Rome
3  New City
WML – example II

```html
card id="card_two" title="Pizza selection">
do type="accept" label="cont">
  go href="#card_three"/>
</do>
<p>
... your favorite pizza!
<select value="Mar" name="PIZZA">
  <option value="Mar">Margherita</option>
  <option value="Fun">Funghi</option>
  <option value="Vul">Vulcano</option>
</select>
</p>
</card>
card id="card_three" title="Your Pizza!">
<p>
Your personal pizza parameter is $\{PIZZA\}$!
</p>
</card>
</wml>
```

WMLScript

- Complement to WML
- Provides general scripting capabilities

- Features
  - validity check of user input
  - check input before sent to server
  - access to device facilities
  - hardware and software (phone call, address book, etc.)
  - local user interaction
  - interaction without round-trip delay
  - extensions to the device software
  - configure device, download new functionality after deployment

WMLScript Structure

```plaintext
Function currencyConvertor(currency, exchangeRate) {
  return currency*exchangeRate;
}

Function myDay(sunShines) {
  var myDay;
  if (sunShines) {
    myDay = "Good";
  } else {
    myDay = "Not so good";
  }
  return myDay;
}
```
WMLScript - example

```javascript
function pizza_test(pizza_type) {
  var taste = "unknown";
  if (pizza_type = "Margherita") {
    taste = "well... ";
  } else {
    if (pizza_type = "Vulcano") {
      taste = "quite hot";
    }
  }
  return taste;
}
```