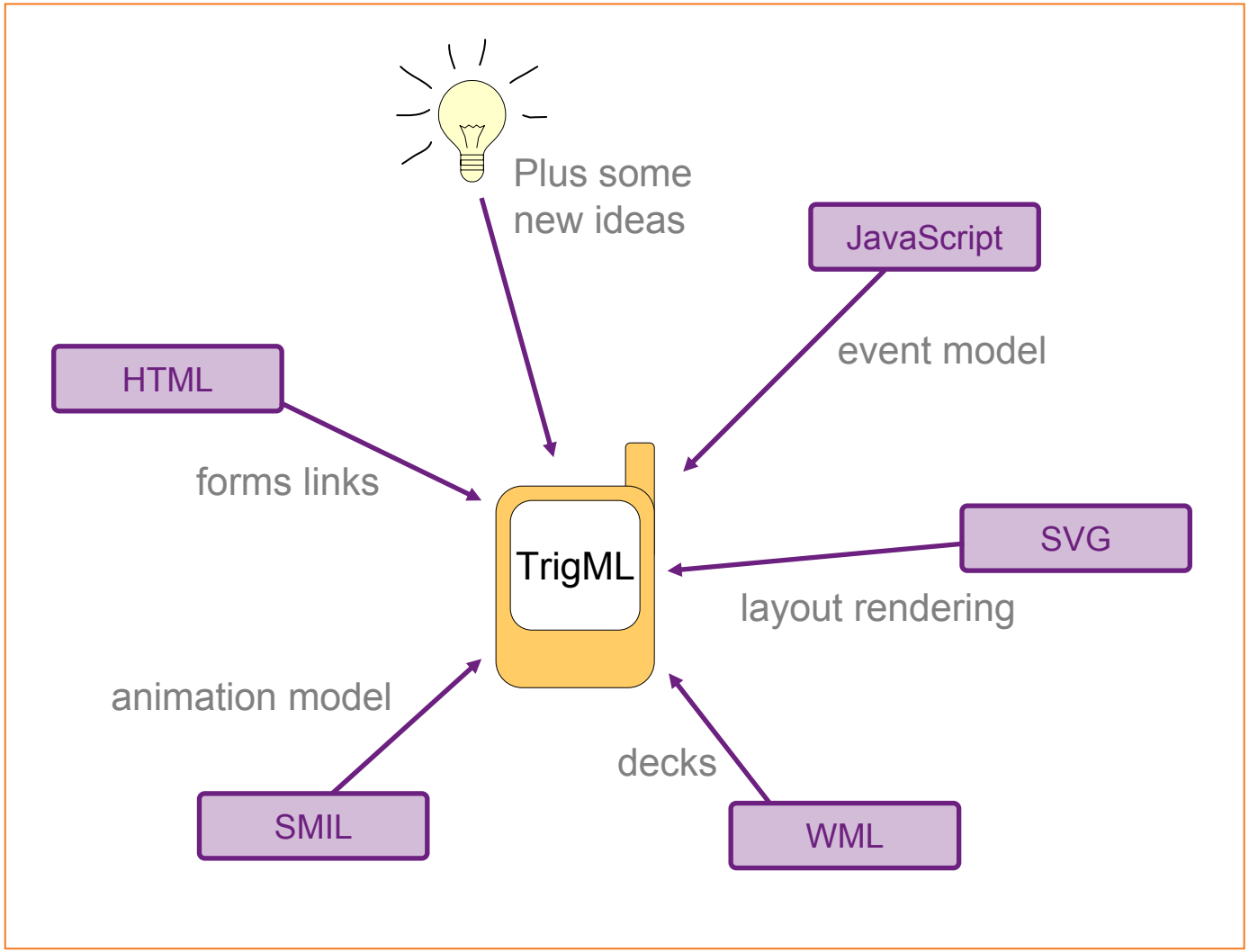
uiOne in a Nutshell

uiOne is user interface platform that enables dynamic user experiences on mobile devices

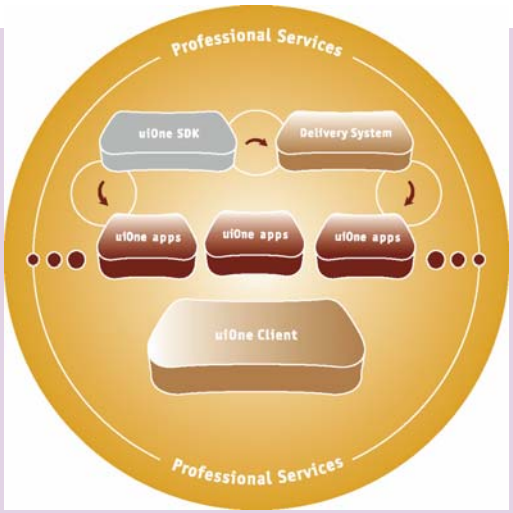
```
<trigml>
  <layer id="layer1">
    <group x="left" y="top" width="*" height="20">
      <image res="banner/logo"/>
    </group>
    <grid rows="3" repeatover="news/headlines">
      <group>
      </group>
    </grid>
    <group x="left" y="bottom" width="*" height="20">
      <image res="footer"/>
```

TrigML® is optimized for Mobile UIs

TrigML provides simple yet powerful language for layout and UI flow



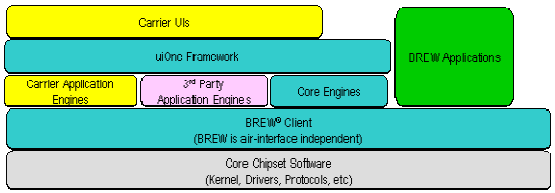
Four Key Components for UI Customization



- **uiOne SDK (Software Development Kit)**
  - Help break UI development into manageable parts and simplify the customization and modification process
  - Authoring tools for the creation of a customized UI including development, debugging, and UI production process management capabilities
  - Device software components necessary to run dynamic and rich UIs
- **uiOne Applications, Resources and Templates (ART)**
  - Global library of extensible UI application resources and design templates created by engineers and visual and interactive designers for use by device manufacturers and operators
- **uiOne Delivery System**
  - Enables ongoing "pushes" of content for UI updates, UI themes, and UI creation, as well as bundled user experience service packages, ultimately creating a personalized, dynamic UI
- **QUALCOMM Professional Services**
  - Team of experienced UI software engineers and user experience designers are available to address any area of system and product customization

A complete UI Solution

uiOne



uiOne Resides on the BREW® Client

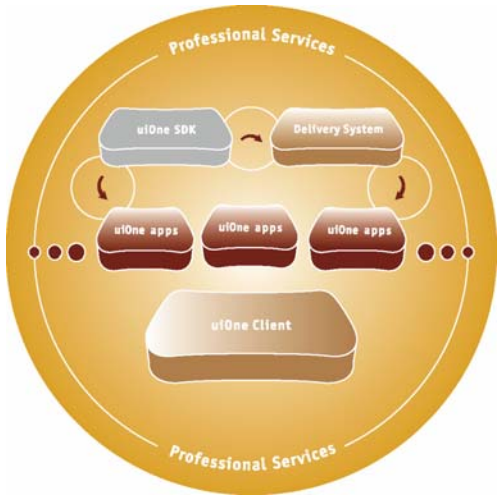
Components of deliveryOne™

uiOne Publisher

Used by operators and/or aggregators to publish content to a content server.

uiOne Depot

Software on a server that creates a storage depot for uiOne content.



Tools & Services

Dynamic Homescreen

Communicate promotion and service information into the idle screen with direct links to shopping areas

Theme Selection

Extends deeply into device screens

mShop

Simple, intuitive shopping experience

Solution Goals of uiOne

- **Enabling operator UI customization**
  - Differentiate brands and services via rapid access and high usability within the UI
  - UI's ability to converge and spotlight features
- **Promoting consumer personalization**
  - Users can make major personalization changes to their phones through one simple step
  - OTA downloadable/updateable
- **Reducing OEM time to market**
  - Realistic approach for OEMs to meet both operator and consumer needs
  - While empowering OEMs to differentiate their own UIs and device personalities
- **Enhancing developer creativity**
  - Expand into new UI application development marketplace previously reserved for device OEMs
  - Create an expansive UI-theme market consisting of rich and over the air (OTA) updateable content

Carrier Customized UI

- **Consistent user experience across phones**
- **Expression of brand throughout the UI**

Home Screen

Main Menu

Favorites

infoCast

Messaging

Shopping Portal

End-User Personalized UI

- **Monetize personalization phenomenon**
- **Increase end-user emotional attachment to device**

Home Screen

Main Menu

Favorites

infoCast

Theme Management

Shopping Portal

uiOne Offerings

mShop

A locally stored, dynamic portal controlled by the operator

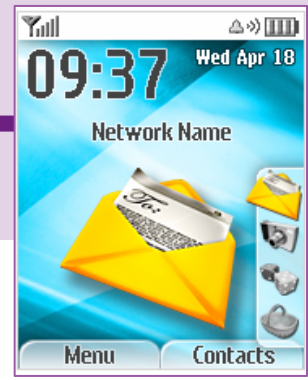
OTA update opportunities throughout UI

Instant access to shopping, no WAP portal access delays

- **Dynamic and interactive shopping experience for BREW applications using deliveryOne or marketOne**
- **Fully updateable and “always on” storefront to promote mobile content and applications**
- **Traditional catalog browsing capability with a rich interface**
- **Multiple areas to promote special applications, create microstores, and target the shopping experience to specific market segments**

Homescreen

- **Homescreen**
  - Customizable template, with simple default implementation
  - Supports softkey and header area decoration, etc.
  - Supports time/date and banner text
  - Supports push of announcements, both text and graphical*
  - OEM wallpaper integration*
- **Main Menu**
  - Customizable template, with simple default implementation
  - Ability to launch uploaded BREW and native apps
  - OTA updateable links via addition of update channels*



*Homescreen 1.1 feature (available 2H 2006)

Dynamic UI Personalities (Themes)

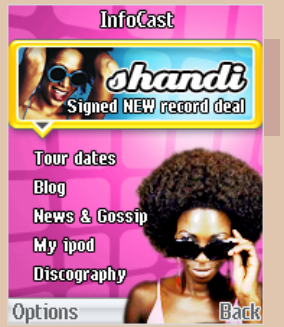
Change the look and layout of the device UI

The layout changes how the device is used!

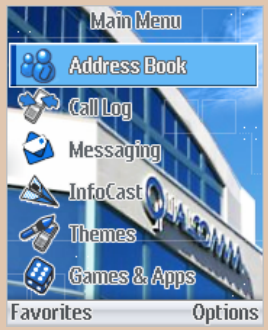
Personalities support an operator's different segments

OTA Themes can apply to each personality

More apps & services



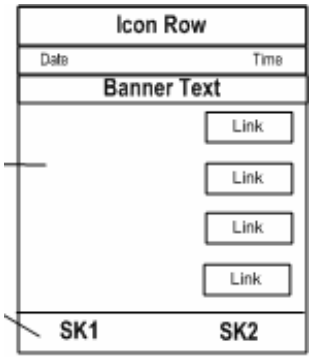
More apps & services



More apps & services

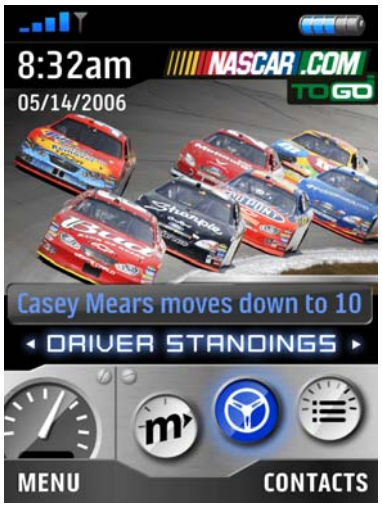
More Themes, Personalities, & Segments

Example Themes



Template-based Homescreen Themes

"Open Canvas" Homescreen Theme



"Live" Theme

Homescreen, Main Menu and Favorites Theme

InfoCast*

Receives pushed and scheduled multimedia info

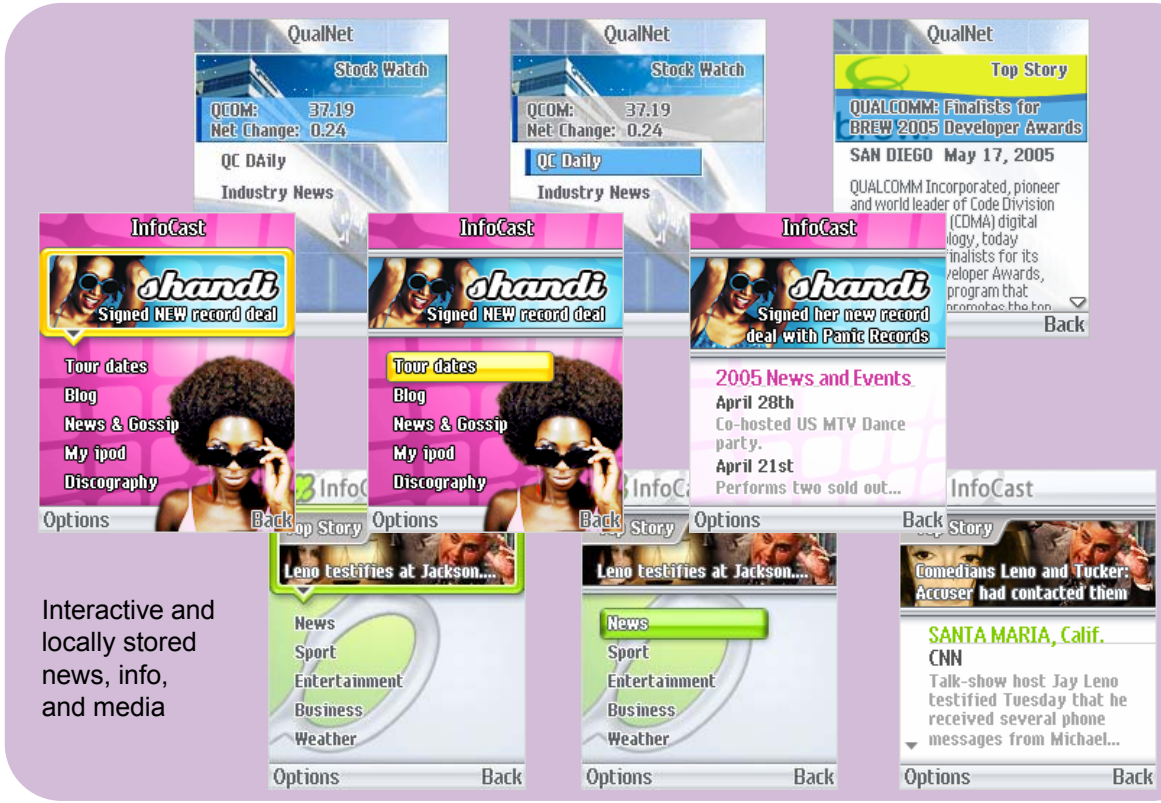
Locally cached for immediate access

Adaptable to different Themes and Personalities

Can be applied to multiple areas across the device UI

*Tentative 1H 2007 availability

Active Homescreens



Interactive and locally stored news, info, and media

“Glance” *

“Glance” is a new way to expand the real estate on the Homescreen and allows preloaded or downloaded “modules” to register with the Glance feature and display a compact UI on the left or right panels of the Homescreen.

*Tentative 1H 2007 availability

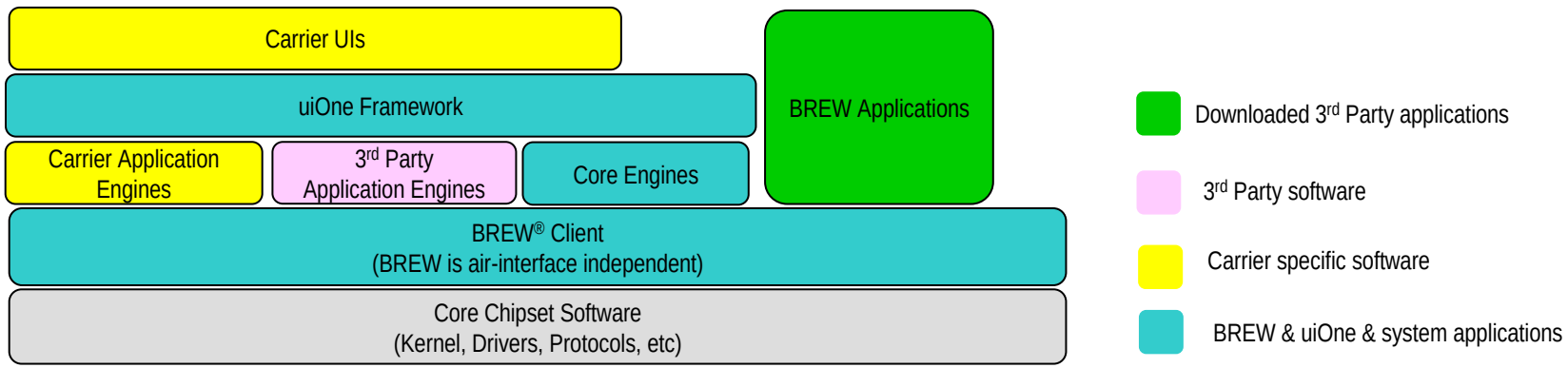
Glance Left

Glance Right

uiOne Technology

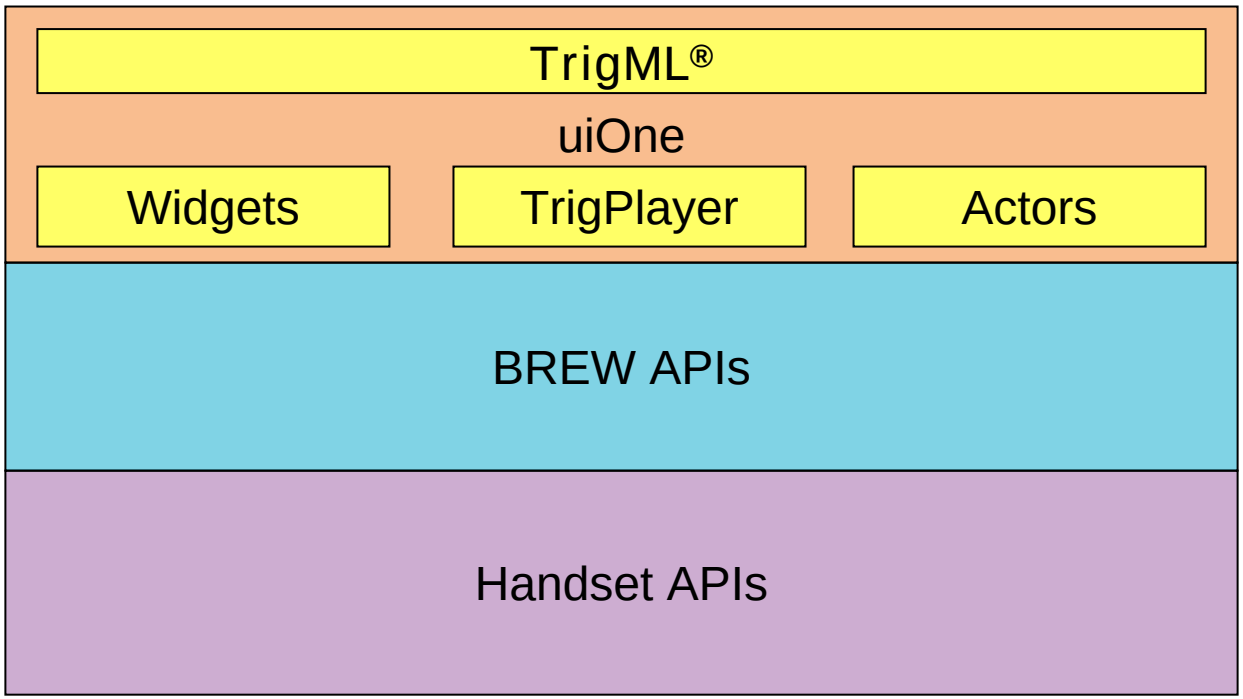
uiOne is a Richer, Broader UI Experience

- **uiOne is a platform that lets developers "dig deep" through BREW Client's porting layer**
  - Providing access to core device capabilities
  - Enabling richer UIs
  - Ensuring cross handset consistency
- **BREW and uiOne fully integrated to wide range of MSM chipsets**
  - Reducing device integration effort and optimize time-to-market
- **Through deliveryOne™, operators have the ability to securely update any components in the UI software stack**



uiOne Technology

- **Driven by QUALCOMM's Binary Runtime Environment for Wireless® (BREW®)**
- **uiOne supported in BREW 3.1.4 or higher**

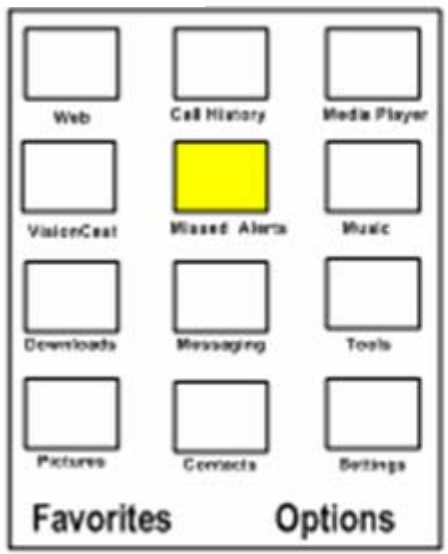


uiOne Logical Architecture

uiOne Technology (Contd.)

- uiOne Themes utilize Trig, Triglets and TrigML
- A Trig provides a logical UI framework (pages, navigation, default resources)
- A Triglet extends a Trig, providing the theme resources (additional logic, layout, images)

Trig



Trig = Logical UI Framework

Triglet

Triglet = Theme Content

uiOne Technology (Contd.)

TrigML Example

Adapt to display size…

…or set exactly

Flexible, easy to use, layout model

```
<trigml>
  <layer id="layer1">
    <group x="left" y="top" width="*" height="20">
      <image res="banner/logo"/>
    </group>
    <grid rows="3" repeatover="news/headlines">
      <group>
        <image res="icons/bullet">
          <att when="focus" name="res"
                            value="icons/selected"/>
          <load when="keypress[select]"
                res="news/headlines/$$/more"
                target="layer2"/>
        </image>
        <text res="news/headlines/$$/title"
              font="$newsfont"/>
      </group>
    </grid>
    <extn:myWidget x="right" width="*"/>
  </layer>
  <layer id="layer2"/>
</trigml>
```

Templated list definition provides compact yet full control over style and appearance

Event model controls focus appearance…

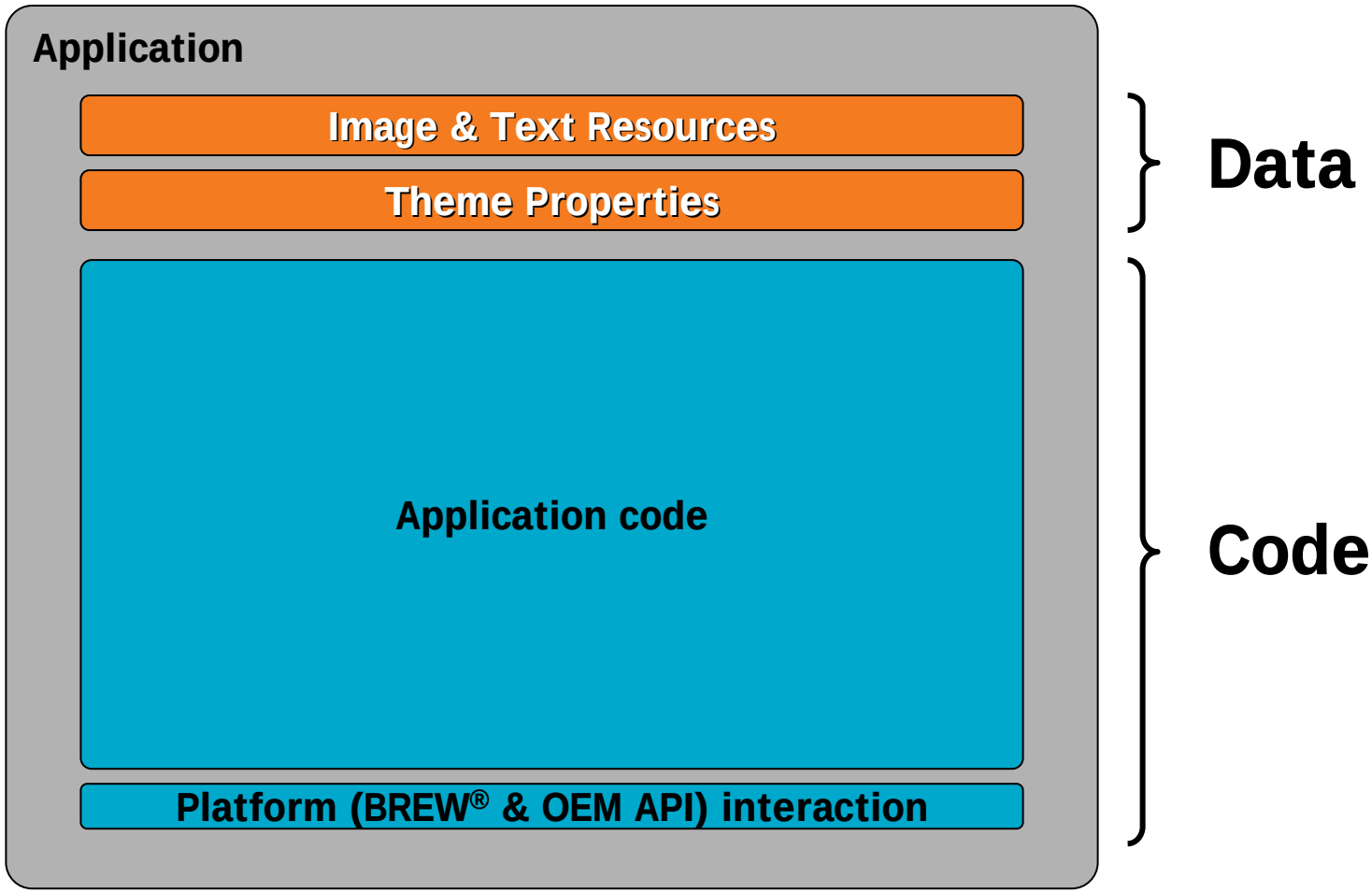
…and page loading

Variables for flexibility and ease of maintenance

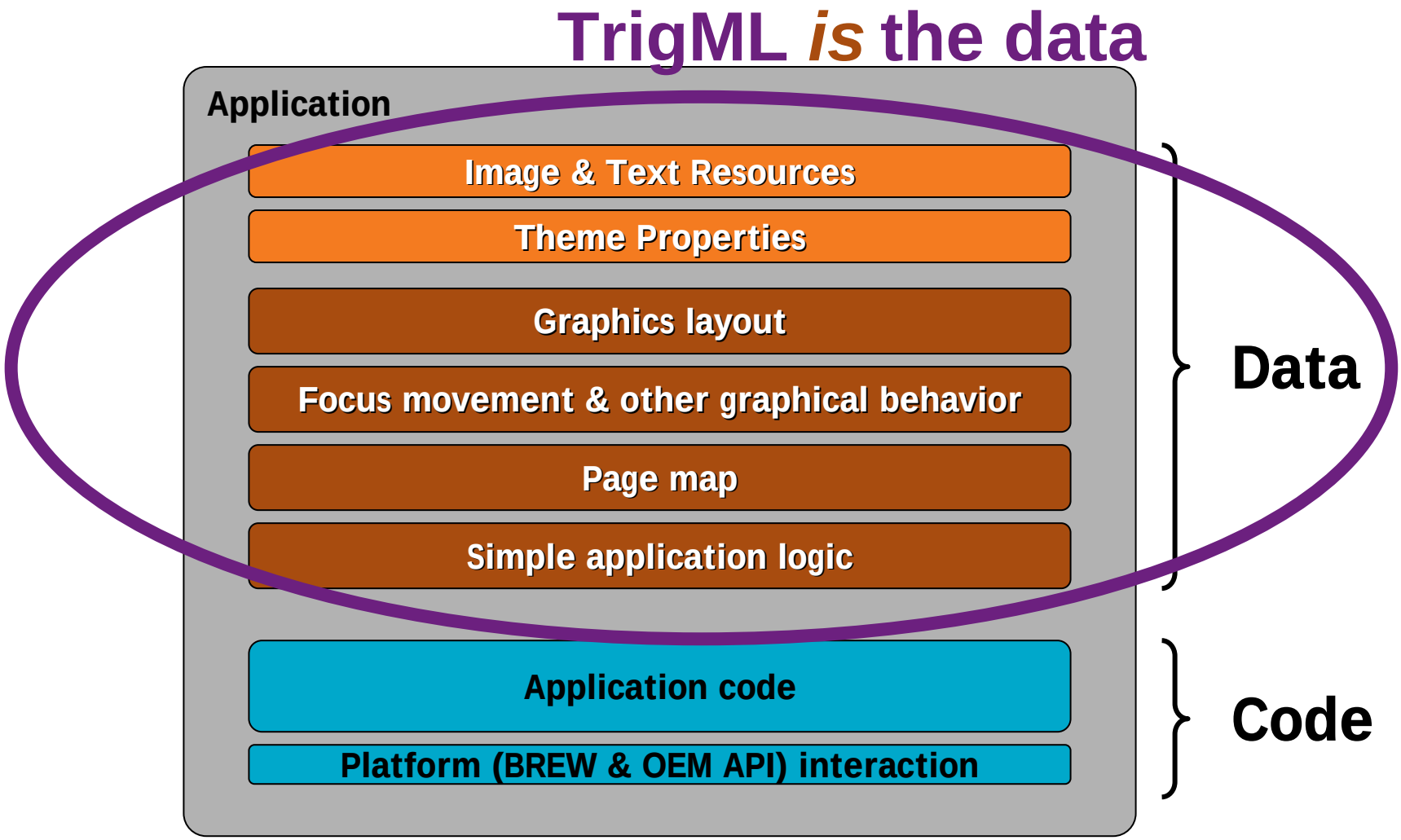
Extend with new widgets

Layering model provides for multiple contexts

Traditionally, data supports UI

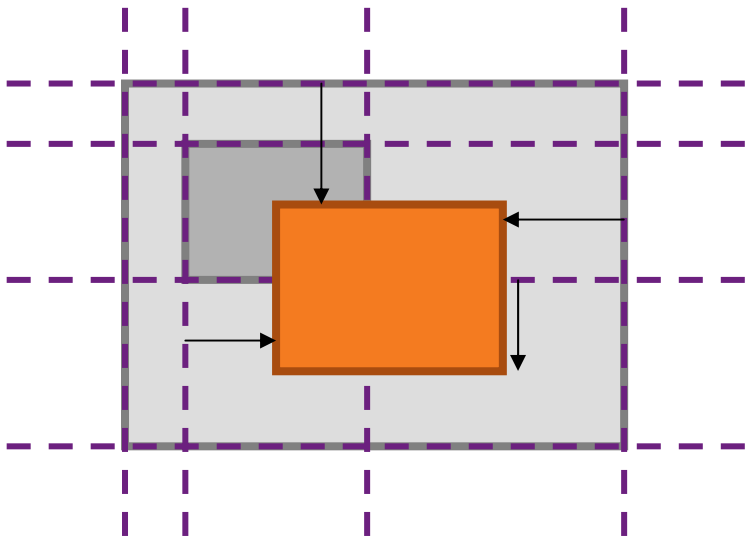


With uiOne™, data *is* the UI



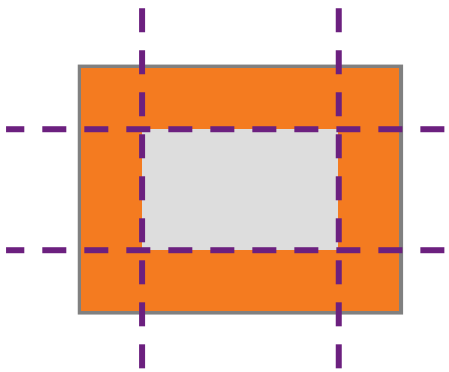
New attributes help keep content flexible

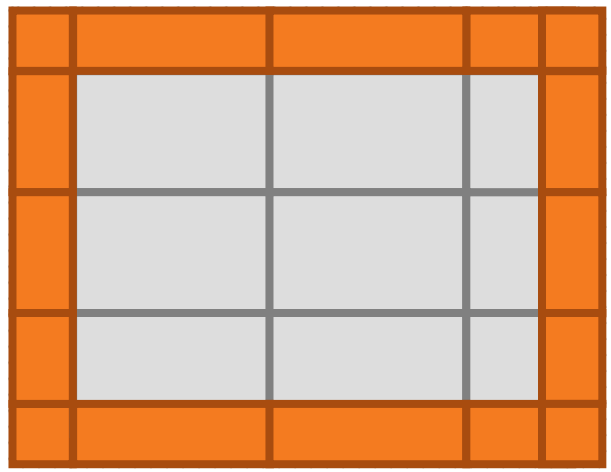
- **‘left’, ‘right’, ‘top’ and ‘bottom’**
  - individually define the positions of edges
  - edges can be set relative to parent or previous element
  - avoids need for excessively nested groups and grids



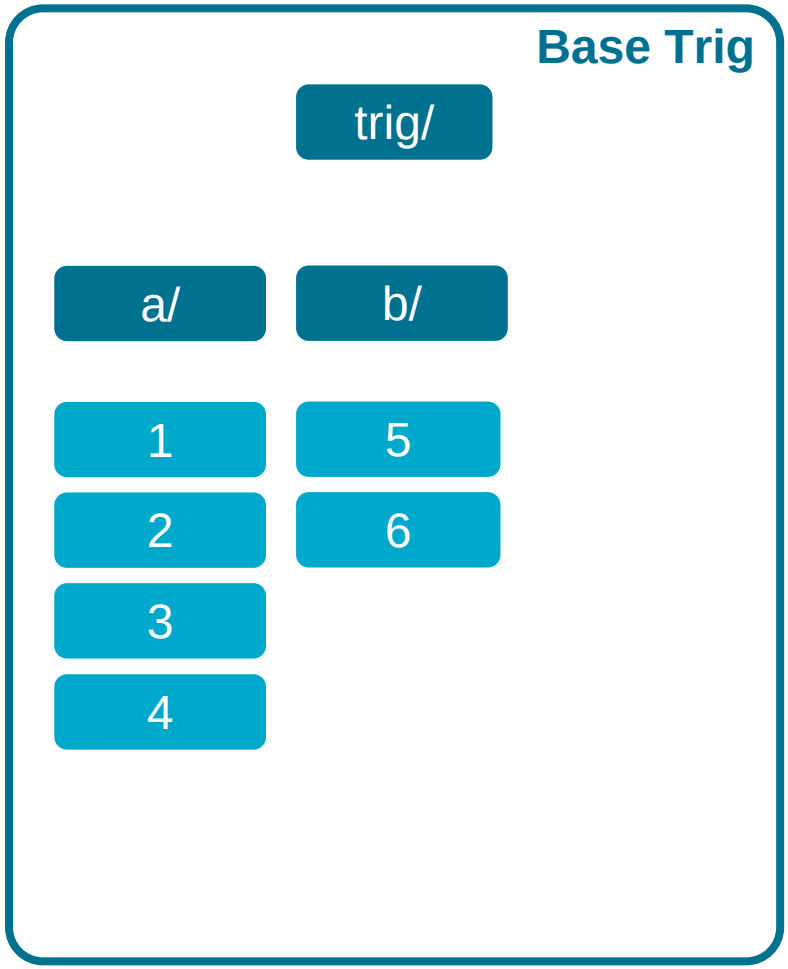
Pixel-perfect image scaling with <tile>

- Re-use a single image resource at many sizes
- “9-slice” the image to preserve borders
- Use shrink-to-fit and edge-control for adaptive sizing of backgrounds





Trig Inheritance



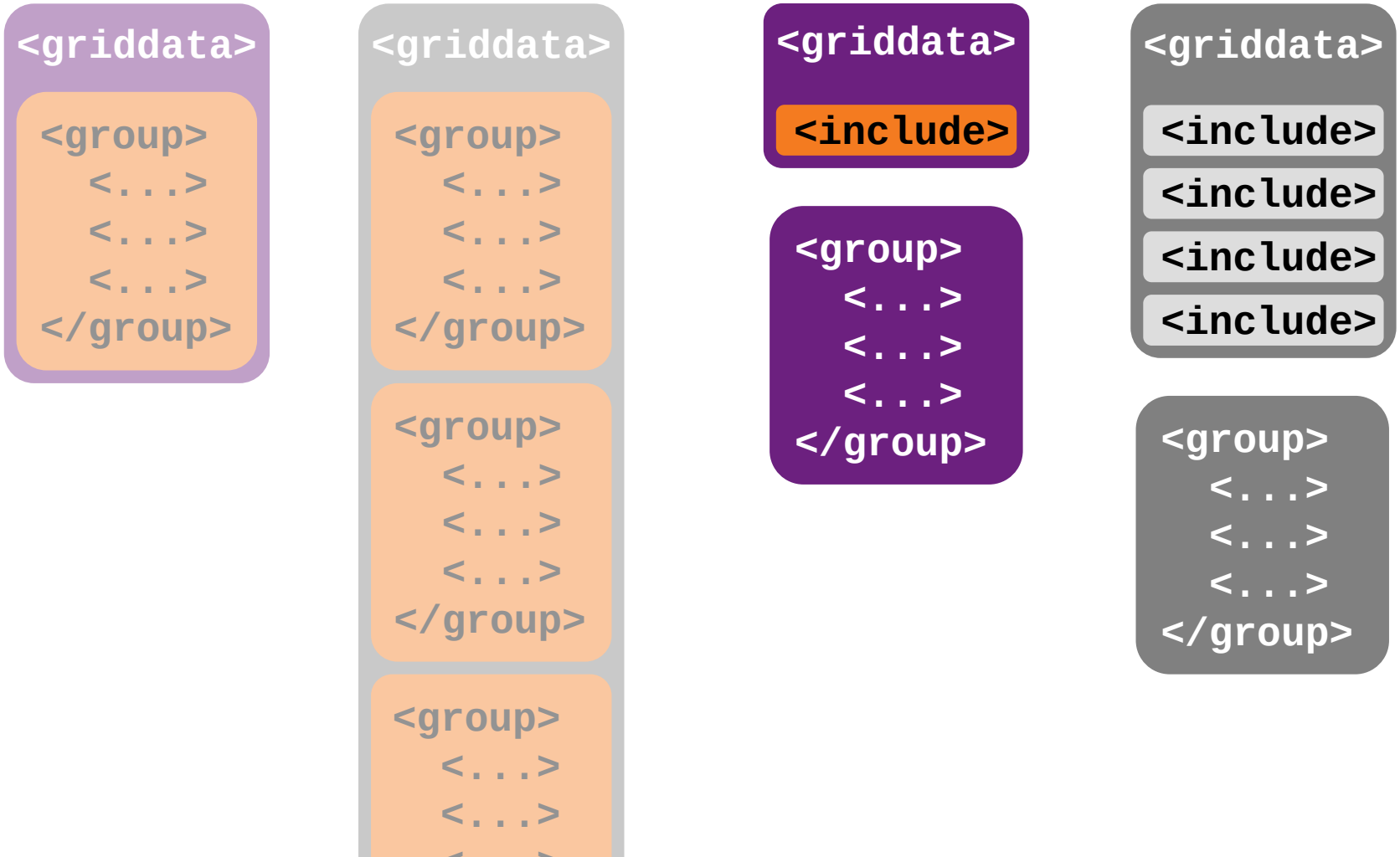
- **Base Trig**
  - Page management utilities
  - skeleton containers
- **Page Trig**
  - page definitions
  - page navigation
- **Theme Trig**
  - branding
  - backgrounds
  - icons and highlights

Derived Trig
trig/
a/
b/
c/
1
2
3
4
5
6
7
8
9
10
11
12
13
14

Data iterators

- **<group> also has a 'repeatover' attribute**
  - Acts as a data iterator or single-cell <griddata>
  - Control iterator with *advance* and *reverse*
- **Useful for tickers and item-by-item browsing**
- **Combine with <anim> for animated transitions**

<include> in <griddata> for smaller Trigs



Integrated Video

- **New <video> tag**
  - Builds on BUIW FrameWidget
- **Video feeds from Actors**
  - ‘res’ attribute points at VFS path
  - VFS path expected to supply an IFrameModel
- **Feeds possible from any source**
  - Viewfinder
  - Camcorder
  - Video resource playback
  - Video streaming
- **Can be layered with other elements**

  (device performance permitting!)

Actor Development Topics

- **What are Actors? What can they do? Where do they live? What comprises an actor? Etc.**
- **Communication Paths**
  - Trig → Actor
  - Actor → Trig
  - Actor → BREW
  - BREW → Actor
  - Actor → Actor
- **Context Issues**

What are Actors?

- **The layer between TrigML and BREW**
  - They enable mark-up to access phone functions via the BREW SDK®
- **Are BREW extensions**
- **Implement IActorLoader and IVfsNode**
- **Enable TrigML to control the phone**
- **Analogy:**
  - TrigML = HTML
  - Actors = HTTP URLs

Why Do I need actors?

- **Can’t express what’s needed in TrigML alone**
- **Need access to BREW APIs to make an application function**

What Can Actors Do?

- Publish and/or supply notifications about data
- Export VFS nodes with arbitrary data/trees
- Throw events at other VFS nodes
- Throw events to Trig
- Respond to events thrown from other entities

Actor Construction

- **Actors:**
  - Constructed at run-time (nodes, containers, etc…)
  - Loaded on-demand
  - Are garbage-collected if unreferenced

Actor IDs

- **String name of actor mapped to class ID**
- **Example:**

  <MimeType Base="0x0102e1f7" Handler="0x01031687“ Value="x-qcactor/callservices"/>

  – 0x01031687 is the callprocessingactor’s class ID (callprocessingactor.bid)

  – 0x0102e1f7 is the ActorLoaders class ID (actorloader.bid)
- **TrigPlayer does a lookup (ISHELL_GetHandler)**

At Construction Time, Actors are:

- **Supplied a pointer to IActorContext, used to:**
  - Find the root node “/”, which they use to access:
    - /trig
    - /var
    - /actor/<actor_name>
    - /elem/<element_ID>
  - Obtain the applet’s class ID
  - IActorContext is the factory for IVFSNodes/IVFSContainer

TrigML - Actor Communication and Control

- **Inbound Event Usage**
  - Example: making a phone call:

```
<throw event="MakeCall" target="/actor/callservices">
    <param name="number" value="8675309" />
    <param name="statusNode" value="/var/origStatus" />
</throw>
```

TrigML - Actor Communication and Control

- **Construct the actor**
  - TrigPlayer looks up the class ID
  - Actor is instantiated
  - Actor receives the event

TrigML - Actor Communication and Control

- **New function sets the event handler**

```
CALLSERVICESCDMAACTORMODEL_VTBL(pMe)->HandleNodeEvent =
  CallServicesCDMAActorModel_HandleNodeEvent;
```

- **(Field Access Macro):**

```
static __inline AEEVTBL(IVfsNodeModel) *
  CALLSERVICESCDMAACTORMODEL_VTBL(CallServicesCDMAActorModel *pMe)
  {
     return AEEGETPVTBL(pMe,IVfsNodeModel);
  }
```

TrigML - Actor Communication and Control

- **Handle event function prototype:**

```
boolean CallServices_HandleNodeEvent
        (IVfsNodeModel *po,
         IVfsNode *node,
         AEEEvent event,
         uint16 wParam,
         uint32 dwParam);
```

- **This is an example of IVFSNODEMODEL_HandleNodeEvent**

TrigML - Actor Communication and Control

- **Event handler determines the event:**

  TrigmlEvent const *trigEvt = (TrigmlEvent const *) dwParam;

  See if: trigEvt->eventName

  is: “MakeCall”

- **Note: don’t use unbounded string manipulation functions, for example use WSTRNCMP not WSTRCMP**

TrigML - Actor Communication and Control

- **TrigML Event Parameter list received by handler:**
  - Arbitrary length
  - Null-terminated list of TrigmlParam structs (dwParam)
  - See: AEETrigmlEvent.h
- **Handler function calls into BREW:**
  - For example: ICALLMGR_Originate(etc, etc…..);
- **Then sets the status:**
  - Use IVFSNODE_SetData(etc, etc…)

TrigML - Actor Communication and Control

- **Nodes**
- **Example: control the vibrator via an integer node**
- **TrigML:**

```
<setdata when="_entry" res="/actor/vibe/isEnabled"
  value="_true" />
```

TrigML - Actor Control Via Integer Node

- **Construct the actor**
  - TrigPlayer looks up the class ID
  - Actor is instantiated
- **New function sets the write function:**

```
VIBEACTORMODEL_VTBL(pMe)->SetNodeData =
  VibeActorModel_SetNodeData;
```

- **VIBEACTOR_MODEL_VTBL is:**

```
static __inline AEEVTBL(IVfsNodeModel) *
  VIBEACTORMODEL_VTBL(const VibeActorModel *pMe)
{
  return AEEGETPVTBL(pMe,IVfsNodeModel);
}
```

TrigML - Actor Control Via Integer Node

- **Write Function prototype:**

```
int VibeActorModel_SetNodeData(IVfsNodeModel *po,
                               IVfsNode *node,
                               AEECLSID type,
                               const void *pObj,
                               int size);
```

- **This is an example of IVFSNODEMODEL_SetNodeData**

TrigML - Actor Control Via Integer Node

- **Index through the nodes, call a BREW extension:**

```
ISOUND_Vibrate(pMe->pISound,VIBE_ON_DURATION);

(void)ISHELL_SetTimer(
   pShell,
   VIBE_RENEW_INTERVAL,
   VibeActorModel_VibeRenewalTimerCB,
   (void *)pMe);

pMe->isVibeOn = TRUE;
```

TrigML - Actor Communication and Control

- **Which is preferable: node or incoming event?**
  - Events package parameters as a unit
  - A node is a single entity
  - Nodes are synchronous; processing is immediate
  - Events are asynchronous (queued) and hence take up more overhead (e.g. memory allocations)
- **Beware:**
  - Because of the synchronous/asynchronous difference, unexpected behavior can occur when you mix the two

Actor - TrigML Communication – Outbound Events

- In general avoid events targeting “_root”, due to focus/event routing and system performance issues
- Any TrigML that doesn’t have focus/is not in the chain of focus will not see the event
- Use to synchronize behavior?
- Can package multiple bits of data
- Can capture data at one point in time
- Prefix all outbound actor events with the actor's string name
- Are queued

Actor - TrigML Communication - Nodes

- Trig listens on actor VFS nodes
- When info arrives, the actor notifies nodes
- If no TrigML is listening, event processing stops
- Data is delivered as a unit (datum); multiple nodes might be needed for one logical event
- Multiple updates to the same node could overwrite unprocessed data
- Nodes are not queued

Actor - TrigML Communication - Nodes

- **Example: battery reaches power-down level**
  - Battery actor's status node changes to '1'

```
<throw when="[{/actor/battery/status} == 1]"
   event="SendURL" target="/actor/shell">
   <param name="url" value="powerdown:Top" />
</throw>
```

Actor - TrigML Communication - Nodes

- **Example: Trig passes a status node path as a parameter to an inbound event:**

```
<throw event="MakeCall" target="/actor/callservices">
   <param name="number" value="8675309" />
   <param name="statusNode" value="/var/origStatus" />
</throw>
```

  - The origination attempt fails (or succeeds)
- **Optionally: the actor publishes a well-known status node**

Actor - TrigML Communication - Nodes

- **Listener Trig responds to the node notify and throws an event to a system notification actor**

```
<throw when="[({/var/origStatus} gt 0) and ({/var/origStatus} lt 100)]"
        event="Add" target="/actor/sn">
      <param name="name" value="callOrigFailedNote" />
      <param name="priority"
       value="{type2notes/callOrigFailedNote/priority}" />
      <param name="unique" value="_true" />
      <param name="origStatus" value="{/var/origStatus}" />
  </throw>
```

Actor - BREW Communication and Control

- **BREW calls**
  - Example – call origination:

```
result = ICALLMGR_Originate(
    pCallMgr,
    dialCallType,
    dString,
    NULL,
    &pCall,
    NULL);
```

BREW - Actor Communication

- **EVT_NOTIFY**
  - Set up in two places:

    1) (Assuming one trigplayer per applet): in the Applet's MIF file e.g.:

    <Notification Notifier="0x01001051" Mask="0x0000003f"/>

    BREW knows to send these events to this applet

BREW - Actor Communication

- **EVT_NOTIFY setup continued**

  2) In the trig/config area (use TrigBuilder), e.g.:
  config/actors/callstate/notifiers/IPhoneNotifier

  is an (arbitrarily named) text element that contains:

  0x01001051

  This tells the TrigPlayer: ‘if you see this notifier, route it to this actor”

- **Note: must be done for every Trig that uses the actor**

BREW - Actor Communication

- **EVT_NOTIFY setup continued – Trig XML file output:**

```
<resource path="config/actors/callstatecdma/notifiers/IPhoneNotifier">
    <text-resource>
      <resource-info name="IPhoneNotifier for the CallStateCDMAActor">
        <description/>
        <status/>
      </resource-info>
      <instances>
        <data-instance>
          <dev-lang-criteria lang="xx" dev="all"/>
          <data ext="txt" format="text">0x01001051
          </data>
        </data-instance>
      </instances>
    </text-resource>
  </resource>
```

BREW - Actor Communication

- **EVT_NOTIFYs impact actor construction**
  - If the actor isn't created (no references):
  - Player creates the actor, sends it the event, then deconstructs it
- **Other BREW events configured in TrigML, example:**
  - config/actors/keyactivity/eventmask → .xml file
  - But not setup in MIF file

Actor - Actor Communication and Control

```
IVfsContainer *pRootNode = NULL;
VfsPathElement path[4];
path[0] = L"actor";
path[1] = L"vibe";
path[2] = L"isEnabled";
path[3] = NULL;

result = IACTORCONTEXT_GetRoot(pContext, &pRootNode);

if (SUCCESS == result) {
   result = IVFSCONTAINER_SetData(pRootNode,
                                  path,
                                  AEEIID_VFS_INTEGER_DATA,
                                  (void*)pData,
                                  sizeof(int));
}
```