Dynamix: An Open Plug-and-Play Context Framework for Android

Darren Carlson and Andreas Schrader
Ambient Computing Group / Institute of Telematics
University of Lübeck, Germany
www.ambient.uni-luebeck.de

This presentation includes graphics from the Crystal Project, which is released under the LGPL license.
Motivation 1/2: The Explosive Rise of Mobile Computing

High Developer Incentives

- Familiar Languages & Tools
- Mobile App Markets

Improved Device Capabilities

- Powerful Hardware
- Inbuilt Comm/Sensors/Media

Basic Context Sensing

- APIs for accessing Location, Orientation, Sensor & User data, etc.
Motivation 2/2: Context-awareness Challenges Remain

Advanced Context Sensing and Acting

- Unproxied Sensor Networks
- Biotelemetry Data
- Indoor Positioning
- User Activity
- Social Proximity and Networks
- Ad-hoc Interactions
- External Sensors
- Sensor Fusion
- Others…

Wide-area Context Infrastructure

- Instrumentation scalability
- Multiple administrative domains (physical and virtual)
- Context sources/actuators not known at design time
- Lack of adaptive context middleware for mobile scenarios
Introducing Ambient Dynamix

Dynamix is a plug-and-play context framework that helps applications sense and adapt to the user’s continuously evolving situation and requirements.

**Android Devices**

Dynamix provides **simple means** for apps to request context support.

Dynamix **adapts the user’s device** to the environment **using plug-ins**.

Includes a **scalable infrastructure** for sharing plug-ins.

Plug-ins are provisioned to the device **at runtime** (network or file system).

Domain experts create **context plug-ins**.

**Community-based approach** with 3rd party API support.
Overview of the Dynamix Framework

Context domain expertise is packaged into sharable plug-ins

A Dynamix app requests context support

Plug-ins are dynamically provisioned at runtime
Dynamix Framework Features

- Runs as a lightweight **background service** on a user’s unmodified Android-based device.
- Apps use simple **Facade and Event APIs** to request context support and receive context events.
- Performs context interactions using a tailored set of plug-ins, which are **dynamically provisioned** to the device during runtime (*from the network or local file-system*).
- Supports **ad-hoc interactions** with discovered resources.
- Sends context information to apps using plain old Java objects (**POJOs**) or string-encoded formats.
- Supports parallel plug-in installations, automatic updating, event caching, and power management.
- Utilizes an **embedded OSGi Framework** to manage Dynamix plug-ins internally.
- Features a **Plug-in Security Sandbox**, which provides managed access to sensitive services and resources.
Open Community Collaboration

1. Context-domain experts use the Open Plug-in SDK to create Dynamix plug-ins. Plug-ins can be published using public or private repositories.

2. App developers use the Open App SDK to create Dynamix apps. Apps can be deployed from any Android market or elsewhere.

3. The end-user installs the Dynamix Framework once. Users can then download and run Dynamix apps.
Introduction to Ambient Dynamix

Context Plug-in Development Overview

<table>
<thead>
<tr>
<th>Context Plug-in Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUSH</td>
<td>Performs continuous context modeling, while broadcasting context events to all Dynamix listeners holding an associated context subscription.</td>
</tr>
<tr>
<td>No application interaction required</td>
<td></td>
</tr>
<tr>
<td>PULL</td>
<td>Performs individual context scans in response to a Dynamix listener’s requests to do so.</td>
</tr>
<tr>
<td>Application interaction required</td>
<td></td>
</tr>
<tr>
<td>PULL_INTERACTIVE</td>
<td>Same as PULL, but requires user interaction via a custom user interface provided by the context plug-in.</td>
</tr>
<tr>
<td>Application interaction required</td>
<td></td>
</tr>
<tr>
<td>PUSHPULL</td>
<td>Combines both the push and pull functionality, as described above.</td>
</tr>
<tr>
<td>Some application interaction required</td>
<td></td>
</tr>
<tr>
<td>PUSHPULL_INTERACTIVE</td>
<td>The same as PUSHPULL, but pull functionality requires user interaction via a user interface provided by the context plug-in.</td>
</tr>
<tr>
<td>Some application interaction required</td>
<td></td>
</tr>
</tbody>
</table>

Includes base classes for a variety of plug-in types

Developers can release custom data-types as standard JAR file, which are used by app developers.
Dynamix App Development Overview

Apps include a single Dynamix JAR on their build path

**Facade API Method Summary**

- **void** addDynamixListener(IDynamixListener listener)
  Registers the listener to receive Dynamix events.

- **void** openSession()
  Indicates that the calling application wishes to open a session with the Dynamix Service.

- **void** addContextSubscription(IDynamixListener listener, String contextType)
  Adds a context subscription for the specified listener and context type.

- **String** requestContextScan(IDynamixListener listener, String pluginId, String contextType)
  Requests a dedicated context scan using the specified plug-in and context type.

- **void** resendCachedContextEvents(IDynamixListener listener, int previousMills)
  Resends the context events that have been cached for the listener.

**Event API Method Summary**

- **void** onDynamixListenerAdded(String listenerId)
  Indicates that the Dynamix listener has been added.

- **void** onSecurityAuthorizationGranted()
  Notification that the application has been granted security authorization by the Dynamix Service.

- **void** onSessionOpened(String sessionId)
  Notification that a Dynamix session has been opened.

- **void** onContextSubscriptionAdded(ContextPluginInformation plugin, String contextType)
  Notifies the listener that a context subscription for the given context type has been added.

- **void** onContextEvent(ContextEvent event)
  Notification of an incoming context event.

Other events have been omitted for brevity...
Setting Up Context Support

Apps request context support using the Dynamix Service’s Façade API.

1. `dynamix.addContextSubscription(dynamixCallback, "org.ambientdynamix.contextplugins.barcode");`

Apps add context subscriptions for required context types

2. **Barcode Context Plugin**
   An interactive pull-based plug-in that uses the device’s inbuilt camera to detect and decode a wide range of 1D and 2D

   Dynamix downloads and installs associated plug-ins in the background

3. `dynamix.contextRequest(dynamixCallback, "org.ambientdynamix.contextplugins.barcode", "org.ambientdynamix.contextplugins.barcode");`

   If necessary, apps trigger context requests (scans or interactions)
Handling Context Events

Context events are sent to apps using the Dynamix Service’s **Event API**.

Receiving Context Events In the App:

```java
public void onContextEvent(ContextEvent event) throws RemoteException {
    Log.i(TAG, "A1 - onContextEvent received from plugin: " + event.getEventSource());
    Log.i(TAG, "A1 - Event context type: " + event.getContextType());
    Log.i(TAG, "A1 - Event timestamp " + event.getTimeStamp().toLocaleString());
    if (event.expires())
        Log.i(TAG, "A1 - Event expires at " + event.getExpireTime().toLocaleString());
    else
        Log.i(TAG, "A1 - Event does not expire");
    // Check for native IContextInfo
    if (event.hasIContextInfo()) {
        IContextInfo nativeInfo = event.getIContextInfo();
        if (nativeInfo instanceof IBarcodeContextInfo) {
            IBarcodeContextInfo data = (IBarcodeContextInfo) nativeInfo;
            Log.i(TAG, "Received IBarcodeContextInfo with format " + data.getBarcodeFormat() + " and value " + data.getBarcodeValue());
        }
    }
```
Protecting User Privacy with the Context Firewall

Dynamix UI: Context Firewall Management

<table>
<thead>
<tr>
<th>Privacy Risk Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Context information is not personally identifiable and poses a low privacy risk.</td>
</tr>
<tr>
<td>Medium</td>
<td>Context information is not personally identifiable and poses a medium privacy risk.</td>
</tr>
<tr>
<td>High</td>
<td>Context information is potentially personally identifiable and poses a high privacy risk.</td>
</tr>
<tr>
<td>Highest</td>
<td>Context information is likely to be personally identifiable and poses the highest privacy risk.</td>
</tr>
</tbody>
</table>
Apps Can Leverage a Broad Range of Dynamix Plug-ins

<table>
<thead>
<tr>
<th>Context Plug-in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power-aware Location, NFC, Beacon, Orientation, and Photodetector Plug-ins</td>
<td></td>
</tr>
<tr>
<td>ArtNet Plug-in (ad-hoc discovery/control of DMX automation equipment)</td>
<td></td>
</tr>
<tr>
<td>Acoustic Fingerprint Plug-in (Native Code Integration)</td>
<td></td>
</tr>
<tr>
<td>Speech Recognition Plug-in</td>
<td></td>
</tr>
<tr>
<td>Sound Pressure Level Plug-in (Ambient Sound Detector)</td>
<td></td>
</tr>
<tr>
<td>Sleep State Plug-in (Zeo Mobile)</td>
<td></td>
</tr>
<tr>
<td>Heart-rate Biotelemetry Plug-in (Zephyr Sensors)</td>
<td></td>
</tr>
<tr>
<td>OpenSocial Profile Data with Sensor-network Monitoring Plug-in (SmartAssist)</td>
<td></td>
</tr>
<tr>
<td>Barcode Scanner Plug-in (ZXing port)</td>
<td></td>
</tr>
<tr>
<td>Air Quality Monitor Plug-in (Ozone Levels and Pollen Count)</td>
<td></td>
</tr>
<tr>
<td>Weight and BMI Measurement Plug-in</td>
<td>Many more plug-ins in development!</td>
</tr>
</tbody>
</table>
Implementation and Evaluation 1/2

Dynamix Framework
• Comprehensive OSGi-based Android prototype
• Plug-in and App SDKs
• Repository architecture
• Website and documentation
• Tested on many popular Android device types

Plug-in Development
• 15 initial plug-ins (more soon)
• Range of context domains and semantics (push vs. pull)
• Each verified as deployable over-the-air at runtime

App Development
• Six initial prototype apps
• Additional apps are being developed

AmbientWeb Extension
• Exposes full Dynamix functionality to browser-based Web clients
• Drop-in JavaScript libraries
• Includes Wellness App demo, created for the IoT 2012 Challenge

Dynamix Apps
- Dynamix Logger
- Medication reminder system
- Product information and reviews
- Bike Wars! Social exercise app
- Heart rate visualization (Processing graphics engine)
- Ambient campus information
- Sound of the City (in development)
- More soon!
Dynamix exhibits **linear performance characteristics** (CPU and Heap) for typical context scanning rates and event payload sizes.
Help Us Make Dynamix Even Better!

- Dynamix is free, open-source, and looking for contributors!
- Visit ambientdynamix.org for details and documentation.
- To access the developer kits, join the public beta! Email: carlson@itm.uni-luebeck.de for access.
Thank You!

Q&A
Context Plug-in Packaging and Deployment

Sample Eclipse Project

Export

Plug-in OSGi Bundle

Data-type JAR(s)

Plug-in Metadata

Deploy

Create

Open Repository Architecture

Supports public, private and file-system-based repositories (Including Maven repos)
Dynamix from the End-user’s Perspective

The Home Tab
- Authorized Applications
  - Dynamix Logger
    - Tap to setup application privacy...
  - Privacy Policy
    - Blocked
    - The application is blocked from receiving any context information.
  - Custom Privacy Settings
    - Tap and hold any Context Plug-in below to set a custom trust level, or tap it to view its description.
  - Ambient Sound Plugin
    - Blocked (Policy)
  - Barcode Context Plugin
    - Blocked (Policy)
  - Pedometer Context Plugin
    - Blocked (Policy)

The Context Firewall
- Dynamix Logger
  - ID: 10033
- Privacy Policy
  - Highest Trust
  - High Trust
  - Medium Trust
  - Low Trust
  - Blocked

Plug-in installation
- Available Context Plug-ins
  - Ambient Sound Plugin
    - Detects the ambient sound level of the user’s environment using the device’s microphone.
  - Barcode Context Plugin
    - An interactive pull-based plug-in that uses the device’s built-in camera to detect and decode a wide range of 1D codes.
  - Location Context Plugin
    - Provides Android location events using a power-aware version of the Android LocationManager. Supports
  - Pedometer Context Plugin
    - A push-based context plug-in that provides step count and related information.

Android Notifications

Most of the time, Dynamix is invisible to the end-user.
Context Domain Complexity Example (Biotelemetry)

Zephyr BioHarness™ Smart Fabric sensors

Varying Sensor Hardware

Context Processing

Context acquisition and modeling

Business logic

Device control and communication

Service discovery

Protocol handling, synchronization, error recovery

Data pre-processing, feature extraction, quantization, representation

End-user Application

Context acquisition and modeling

Extracted Context Data

• Heart rate
• Temperature
• Breathing rate

Varying Sensor Hardware

Context Processing
Android Integration

- Dynamix operates as a **service** within Android.
- Developers create Dynamix apps using **existing skills and tooling**.
Dynamix and OSGi

- Dynamix uses an **embedded OSGi Framework** as the foundation of its plug-in architecture (Apache Felix).
- Context plug-ins are packaged and **deployed as OSGi Bundles**.
- The **Dynamix OSGIManager** supports multi-threaded Bundle installations and updates; progress notifications; Bundle verification; runtime integration; and plug-in lifecycle management.

![The OSGi Framework Architecture](image)

**Diagram**

- **Facade API**
- **Event API**
- **Context Firewall**
- **Dynamix Manager**
- **Settings Manager**
- **Plug-in Security Sandbox**
- **Capability Manager**
- **Context Manager**
- **Local Hardware / Platform APIs**
- **Driver**
- **Execution Environment**
- **Service**
- **Life cycle**
- **Module**
- **Security**
- **Bundles**
- **Context Plug-ins**
How Context Plug-ins Interact with the User

- Some plug-ins may need to provide user interfaces for configuration or context acquisition (e.g. entering data, pointing a camera, etc.)
- However, Android’s security model requires *preregistration* of Activities
- To overcome this problem, Dynamix provides “Host Activities” that can be dynamically populated with a plug-in’s user interfaces.
Representing Context Information with IContextInfo

<table>
<thead>
<tr>
<th>IContextInfo Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>getContextType()</strong></td>
</tr>
<tr>
<td>Returns the type of the context information represented by the IContextInfo. This string must match one of the context information type strings described by the source ContextPlugin.</td>
</tr>
<tr>
<td><strong>getStringRepresentation(String format)</strong></td>
</tr>
<tr>
<td>Returns a string-based representation of the IContextInfo based on the specified format.</td>
</tr>
<tr>
<td><strong>getStringRepresentationFormats()</strong></td>
</tr>
<tr>
<td>Returns a Set of the supported string-based context representations.</td>
</tr>
<tr>
<td><strong>getImplementingClassname()</strong></td>
</tr>
<tr>
<td>Returns the fully qualified class-name of the class implementing the IContextInfo interface. Used when casting the IContextInfo entity to a concrete implementation.</td>
</tr>
</tbody>
</table>

/*
 * Returns the battery indicator value of the ZephyrHxM device generating this event.
 */

```java
public int getBatteryIndicator() {
    return batteryIndicator;
}
```

/*
 * Returns the heart-rate detected by the ZephyrHxM device generating this event.
 */

```java
public int getHeartRate() {
    return heartRate;
}
```

Example IContextInfo code snippet from a heart-rate monitor plug-in

Developers release custom data-types as a standard JAR file, which are used by app developers.