

# HanPhone Server Design Specification

Last updated: 2006-06-21

## 1. Introduction

This document describes the high-level design of HanPhone Server ("HanPhone"), a platform for developing voice applications that deliver audio information to end users by phone. It describes the design considerations, the high-level architecture and the design of the major modules in more details. It is intended to be a technical overview of HanPhone for anyone who intends to evaluate its design or utilize it for developing voice applications. It is not a functional specification.

## 2. System Overview

HanPhone Server is designed to be a flexible voice application platform that encapsulates computer telephony hardware, TTS (Text-To-Speech) and ASR (Automatic Speech Recognition) software. It allows developers to build voice applications using a subset of HTML or a simple markup language called HanPhoneXML.

HanPhone interprets HTML and HanPhoneXML documents and the carry out actions such as synthesizing text to speech and playing back the audio to the end user, detecting DTMF input by user, make call transfers, etc. according to the instructions in those documents. Input from the user is submitted to predefined URLs in the form of HTTP requests and the responses from the URLs are again interpreted. Figure 1 illustrates this process:

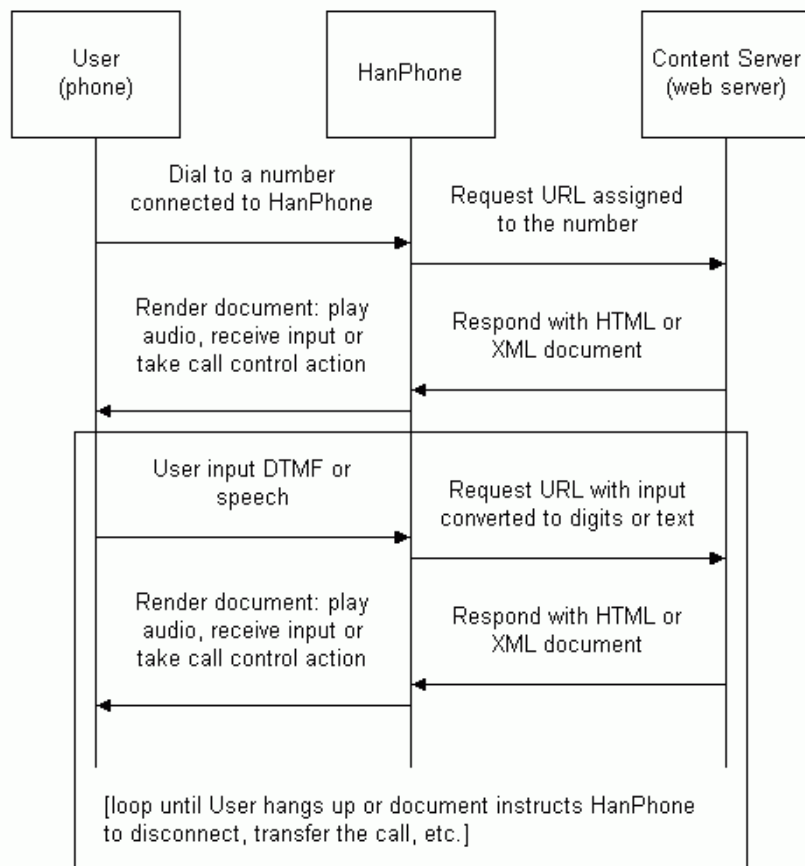


Figure 1 – Sequence diagram of a typical HanPhone session

The entire process is similar to a web browser rendering HTML to the end user visually while accepting input from the user and submitting them to predefined URLs. By adopting a browser-like design, developers can leverage existing web application technologies and experience because the process of developing a call-flow for HanPhone to execute is similar to that of developing a web application.

### **3. Design Considerations**

This section highlights the major design considerations of HanPhone.

#### **3.1. Programming Language**

Java is the language used to implement most of HanPhone because of its following features:

- Object-oriented
- Garbage collection
- Language-level multithreading
- Cross-platform
- Many mature open source class libraries available

In addition, Java is a proven language for developing server-side or server applications. Although its performance is constantly under criticism, it is not a major issue for HanPhone since the bottleneck will be the playback of audio and the retrieval of DTMF or speech input in the telephony hardware.

Only parts of HanPhone that use native (C/C++) code are those that access the C-only APIs of TTS, ASR, telephone hardware.

#### **3.2. System Requirement**

HanPhone is intended to be a cost-effective platform for hosting voice application on average to high-end, server-grade, PCs. The minimal requirements are the following:

- Pentium III 1 GHz or faster CPU
- 512 MB of RAM
- 20 GB free disk space
- OS – Windows 2000 Server, Windows Server 2003
- Java – J2SE SDK 1.4.2
- Dialogic telecom boards supporting analogue or T1 lines
- MySQL 4.1 or above

Note: optional third party software such as TTS and ASR applications require additional free disk space, RAM and CPU processor power.

### **4. System Architecture**

This section describes the overall architecture of HanPhone and some specific details of its major components.

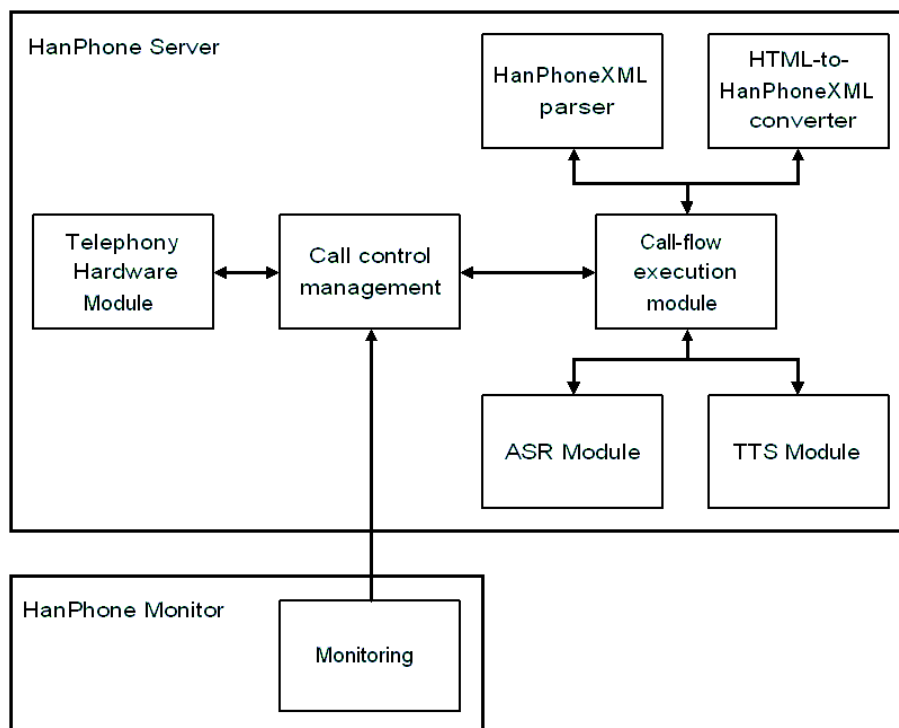
#### **4.1 Overall Architecture**

HanPhone contains modules that specialized in different roles:

- Telephony hardware module

- TTS module
- ASR module
- HTML-to-HanPhoneXML converter
- HanPhoneXML parser
- Call-flow execution
- Call control management
- Monitoring

Figure 2 shows these modules and their relationship:



**Figure 2 – Major modules of HanPhone**

#### **4.2. Telephony hardware module**

This module is the interface between the telephony hardware and the rest of HanPhone. It exposes an API for the other modules to accept incoming calls, retrieve caller ID, playback audio, retrieve DTMF, record speech from the caller, make call transfers, etc.

#### **4.3. Call control management module**

The process of interpreting a HTML or HanPhoneXML begins when this module accepts an incoming call, looks up the index URL associated with the channel or dialed number (DNIS) of the call and applies appropriate filtering to decide if it should hang up on the caller or pass the appropriate index URL to the call flow execution module for interpretation.

This module also records basic call statistics such as caller ID (ANI), DNIS, time of call, duration, channel used, etc.

#### **4.4. Call-flow execution module**

The core of HanPhone is the call-flow execution module. It is responsible for interpreting HTML and HanPhoneXML documents, perform appropriate actions, collect user input, etc. It utilizes the ASR and TTS modules when the document being interpreted calls for speech synthesis out or input.

#### **4.5. HanPhoneXML Parser**

This module parses HanPhoneXML documents and converts their contents into a objects arranged in a tree-like structure internally. The object tree is then traversed by the call-flow execution module.

#### **4.6. HTML-To-HanPhoneXML converter**

The native document format of HanPhone is HanPhoneXML. However, HanPhone also supports a subset of HTML by converting it into HanPhoneXML before interpreting it. The converter makes use of Java's built-in HTML parser and maps the elements and attributes into HanPhoneXML.

#### **4.7. ASR Module**

The call-flow execution module accesses third party ASR application via this module. It provides a unified ASR interface to the call-flow execution module regardless of the actual ASR application loaded.

When speech input is required, the call-flow execution module records any sound coming from the caller into an audio file. Then the audio file is passed to the ASR module for recognition against a pre-defined grammar. If any token defined in the grammar is recognized, the text correspond to the token is returned to the call-flow execution module. It can then proceed to send the text in the form of a HTTP request to the appropriate URL.

#### **4.8. TTS Module**

When the call-flow execution module requires the service of a third party TTS application, it does so via the TTS module. It provides a unified interface regardless of the actual TTS application invoked.

When synthesized speech output is required, the call-flow execution module preprocesses the input text to remove ambiguities first and apply predefined pattern mapping. The preprocessed text is then passed to the TTS module for synthesis. The output is an audio file ready for playback. By default the audio file is cached, i.e. the cached copy of an audio file is fetched the next time the same input text needs to be synthesized.

#### **4.9. Monitoring module**

The monitoring module resides in a separate process canned the HanPhone Monitor. It queries the call control management module to check the status of each call, and analyze calls statistics to determine if there is any abnormal usage. Alerts are sent to configurable email addresses for different abnormal events.