# Service Creation

## Konark - Creating Services

For ease, I will show how to create a service directly in the KonarkAPI. This service class will fit in the "independent services" namespace. There are two parts to creating a service: The XML description and the code the service invoker uses to handle the request. The XML description language used can be read about in more detail in the thesis papers of Nitin Desai and Varun Verma but I will give you an idea of how to write it quickly. The XML description must contain certain tags, this would be part of a standard for Konark. The root tag is:

```
<Service></Service>
```

The second level of tags will always be:

```
<Service>
    <ServiceName></ServiceName>
    <ServiceType></ServiceType>
    <Keywords></Keywords>
    <Duration></Duration>
    <Properties></Properties>
    <Functions></Functions>
</Service>
```

The "service name" will be whatever the developer decides to call it, probably something relating to the content and if this was all standardized, this name would be unique. The "service type" will be a node in the service tree (cannot be changed). "Keywords" is a tag where you can put a list of words that could possibly identify this service. The "Duration" tag is the amount of time in minutes that the service is available for use in the network. "Properties" defines properties by which to describe the service better. "Functions" holds the key to service invocation. It allows the client to choose what kind of service they want and what information they must send to get it.

The next level down for "Keywords" is a variable number of "Word" tags. "Properties" has a variable number of tags "Property" and for each property there is one of each "Name," "Description," and "Value." For "Functions" there is a variable number of "Function" tags each with "Name," "Description," a variable number of "Parameter" tags, and a "ReturnParameter." The "Parameter" tag contains one of each "Name," "Type," and "Description" as does the "ReturnParameter." These describe the input and output to and from client to server. Here is a sample of an XML service description:

```
<Service author="varun verma" Date="Mar 28, 2002">
    <ServiceName>Song for you!</ServiceName>
    <ServiceType>songs</ServiceType>
    <Keywords>
        <Word>songs</Word>
        <Word>music</Word>
    </Keywords>

    <Duration>20</Duration>
    <Properties>
        <Property>
            <Name>Cost</Name>
            <Description>cost of using the service</Description>
            <Value>Free</Value>
        </Property>
        <Property>
            <Name>Format</Name>
            <Description>Format of the music</Description>
            <Value>mp3</Value>
        </Property>
    </Properties>
    <Functions>
        <Function>
            <Name>GetSong</Name>
            <Description>Get the music file</Description>
            <Parameter>
                <Name>file</Name>
                <Type>String</Type>
                <Description>Song to download:</Description>
            </Parameter>
            <ReturnParameter>
                <Name>Value</Name>
                <Type>File</Type>
                <Description>The Music File</Description>
            </ReturnParameter>
        </Function>
    </Functions>
</Service>
```

The next part is writing the code to handle service invocation. This code will be used by the HTTP server to process a request and generate a response. Each service class written will use the "AppServer" interface. One must inherit this interface and simple write the code for:

public byte[] **handleSOAPRequest** ( string **methodName**, ArrayList **inParams**); The method will take the method name in case the service has more than one function and a list of the parameters. Simply write code to process these inputs and return the whole answer as a byte[]. One may also empty the "inParams" ArrayList and fill it with whatever they like. This variable will be available to the service invoker also. When writing a service one should just add a new item in the "File" menu. Make sure this class file is in the Konark namespace and then in the "independentservices" namespace.