BAIKONUR
WEB APPLICATION
SERVER

ADMINISTRATION GUIDE

BAIKONUR Server Administration

The parameters of BAIKONUR server can be changed either with the aid of the ADMIN administration utility or by directly editing the server's system files. The ADMIN utility is described in the "Getting Started" manual. The system files are described in this guide.

NOTE. The ADMIN utility can be utilized to modify only a limited set of the server's editable parameters.

Since the mix of BAIKONUR server administration parameters is fairly extensive, we recommend that you read the section of this guide devoted to the discussion of setting options for various kinds of BAIKONUR server applications.

Starting Up and Stopping BAIKONUR Server

BAIKONUR™ Web App Server can work under Windows 95 or Windows NT (versions 3.51 and 4.0). If used in Windows 95 environment, BAIKONUR server can function only as an application program. In Windows NT environment, BAIKONUR server can function in one of two modes: as a conventional application program or as a service. We recommend that you employ BAIKONUR as a Windows NT service, because only this mode will allow you to control and manage user access rights to system resources. The mode whereby BAIKONUR server runs as an application software can be used for debugging various applications designed to run under BAIKONUR control, as well as for Web site debugging purposes. Regardless of the operating mode, you use one and the same executable file (BAIKONUR.EXE) to run the server.

Starting Up and Stopping BAIKONUR Service

BAIKONUR service is installed automatically when you install the BAIKONUR Web

App Server software package. To run BAIKONUR in the service mode, open the Control Panel and select the “Services” utility there. Find the line “Baikonur” in the list of services and click the “Start” button – this will start BAIKONUR as a service. To stop the service, simply click the “Stop” button.

You can also stop the service by closing its console window (if BAIKONUR’s Log messages are output to the console), preferably by pressing the Ctrl-C key combination.

Initially, the service is started up manually. If you wish to have the service started up automatically every time the operating system is re-booted, click the “Startup” button in the “Services” utility, and check the “Automatic” option.

If you wish to remove the service from your system for some reason, run the program BAIKONUR.EXE with a “–r” switch in the command line, or select the “Remove Service” option.

To install BAIKONUR as a service, run the program BAIKONUR.EXE with a “-i” switch in the command line, or select the “Install Service” option.

Starting BAIKONUR As An Application Program

To start BAIKONUR as a Windows 95 application program, select the “Baikonur for Win’95” option or simply run the program BAIKONUR.EXE. If you work in Windows NT, run the program BAIKONUR.EXE with a “-d” switch in the command line, or select the “Debug Mode” option.

Objects Of Administration

The parameters of BAIKONUR server accessible for administration can be grouped into several categories:

- server’s internal parameters (which condition the way the server will be functioning)
- supported protocols
- sections (virtual servers)
- aliases
- clients (users)
- document types

Server’s Internal Parameters

The internal parameters of BAIKONUR server determine its overall functional logic. These parameters are specified in the [Baikonur] section of the BAIKONUR.INI file, and can be divided into the following groups*.

System Directories and Files Settings

The following parameters help the server determine where to look for the system libraries and system files: *SystemDirectory*, *HomeDirectory*, *HomePage*, *AliasFile*, *ClientsFile*, and *CertificateFile*.

Log File Settings

The following parameters tell the server where (and how) to output its run-time messages: *LogFile*, *LogLevel*, and *LogDirectWrite*.

Time-Related Parameters

The following parameters determine various kinds of timeouts and other time-related events: *TimeResolution*, *TimeClientInactive*, *TimeClientDisactive*, *ApplicationTimeOut*, *ApplicationStartTimeOut*, and *SessionTimeOut*.

Quantitative Parameters

The following parameters determine the maximum allowable number of applications that can concurrently run on the server, the maximum number of connections, the size of send/receive buffers, etc.: *Agents*, *Managers*, *MaxClientConnections*, *MaxClientApplications*, *Applications*, *SocketRcvBuf*, *SocketSndBuf*, and *MaxReceiveBuffer*.

Supported Protocols

BAIKONUR server's functionality can be expanded by connecting additional protocols to it. Apart from the HTTP protocol, the version of BAIKONUR under discussion can also support such protocols as FTP, Gopher and Finger. Auxiliary protocols are made in the form of DLLs and may, if necessary, be added to the already available protocols. The techniques employed to develop DLLs for new protocols are described in detail in the manuals supplied with senior versions of BAIKONUR.

To add a new protocol to BAIKONUR server, you must specify the corresponding settings in the [Baikonur.Protocol] section of the BAIKONUR.INI file, as shown in the following example:

```
[Baikonur.Protocol]
http:80 = HTTP/1.0, HTTP10.DLL, _Parser, On
```

The above sample code denotes that the HTTP protocol parser is assigned to port 80, that the version of this protocol is HTTP/1.0, that the DLL containing the _Parser function is named HTTP10.DLL, and that the parser is active at startup.

A more detailed description of these parameters will be given later in the text (in the section describing the BAIKONUR.INI file).

You can have one and the same protocol parser assigned to several ports, so that it would "listen to" all of them. Thus, apart from port 80 (assigned to the HTTP protocol by default), you can declare an additional HTTP port as follows:

```
[Baikonur.Protocol]
http:80 = HTTP/1.0, HTTP10.DLL, _Parser, On
http:8080 = HTTP/1.0, HTTP10.DLL, _Parser, On
```

The following example illustrates how the FTP protocol can be assigned to port 21:

```
ftp:21 = FTP/1.0, FTP.DLL, _Parser, On
```

When making the assignment(s), see to its that only one protocol parser is assigned to any given port.

The HTTP protocol can be supplemented with the SSL (Secure Sockets Layer) data encryption mechanism. For this mechanism to function properly, you must (a) have the SSL certificates file available, and (b) connect the protocol as shown in the following example:

```
http:443 = HTTP/1.0, HTTP10.DLL, _Parser, On, SSL
```

The path to the SSL certificates file is specified in the *CertificateFile* parameter of the [Baikonur] section of the BAIKONUR.INI file.

Sections (Virtual Servers)

As can be understood from the foregoing paragraph, BAIKONUR has the capability of supporting several protocols, which the user accesses through different ports. However, the server can also be addressed in different ways (by using its symbolic Internet address, the name of the host computer name in the local network, or the numeric IP address). It would therefore be helpful if the server's administrator could somehow differentiate various name-port combinations, and could assign a unique set of parameters (aliases, clients and file types) for each of them. If the administrator wants to have a set of parameters defined for a certain name-port combination, he/she must organize a new section (otherwise known as a virtual server) in the server settings, and specify the necessary settings there.

We recommend that you define a separate section for at least every additional protocol. In this way, you will be able to assign a separate home directory for each of the sections. Otherwise, a situation may occur whereby HTTP and FTP would provide access to one and the same directory, which is not always acceptable.

Virtual servers are defined with the help of the *AliasSection* (*AliasSection0* j *AliasSectionF*) parameter in the [Baikonur.Alias] section of the BAIKONUR.INI file.

For every virtual server in the BAIKONUR.INI file you must specify its name, address-port combination, Administrator and Anonymous users (to be discussed later), authorization scheme, server's response to certain file type requests, and other options. The set of aliases (to be discussed later) is specified in the aliases file.

In the way of an example, let us see how an FTP virtual server can be organized (assuming that the FTP protocol is assigned to port 21).

First, we define a virtual server in the [Baikonur.Alias] section of the BAIKONUR.INI file:

```
[Baikonur.Alias]
. . .
AliasSection=ftp_section(:21)
. . .
```

The above declaration means that whenever an attempt is made to access it through port 21 with the use of any address, the server will employ the settings specified for the section named FTP_Section.

Next, we specify the server's response to requests of certain file types (the type of a file is determined by its extension) in the [Baikonur.Action] section of the BAIKONUR.INI file:

```
[Baikonur.Action]
. . .
*.ftp_section = , SendSource
ini.ftp_section = , AccessDenied
. . .
```

The above declaration means that all files other than files with the INI extension will be sent to the user. The comma to the right of the equals sign is essential (it denotes that the optional *Mime-Type* parameter was omitted).

Then, we go the [Baikonur.Client] section to define the server's clients and the logic employed to access the server:

```
[Baikonur.Client]
. . .
section.ftp_section = , *Guest_id, Basic, On, On, Off,
Off
. . .
```

This definition means that the server is accessible to any Anonymous user (corresponding to the Guest_ID user in the clients file – see below), and that the Basic scheme will be used for user authentication (for details see the section describing the BAIKONUR.INI file).

Next, we can specify the following in the aliases file (ALIAS.FIL):

```
[Alias.Section.ftp_Section]
/ = D:\FTP\Public
info = c:\web\info
```

This means that a client accessing the server using the FTP protocol will have access to the files located in the D:\FTP\Public directory (and its sub-directories). You can define more than one alias in this section.

As far as the HTTP protocol is concerned, the settings of the [Baikonur.Client] section are of considerable importance. In this section you specify whether or not the user should supply a password to gain access to the server, and whether or not the Cookies mechanism (important when working with applications) will be utilized. For details refer to the description of the [Baikonur.Client] section of the BAIKONUR.INI file.

Aliases

BAIKONUR Web server can provide its clients access to the resources (files and programs) located in the server's home directory and its sub-directories. The name of this directory is specified with the aid of the *HomeDirectory* parameter in the [Baikonur] section of the BAIKONUR.INI file.

However, in some cases you would want to allow clients to access resources located in some other directory or even on a different (possibly network) drive. To enable you to do so, we have incorporated a special "aliases mechanism" into BAIKONUR.

Every alias is made to correspond to a certain directory on the server's disk or the accessible network drive. The appropriate settings are specified in a special aliases file (named ALIAS.FIL by default).

From the user's viewpoint, an alias appears as a sub-directory of the server's home directory. Thus, if you assign the alias *info* to the directory D:\NEWS\INFO, the user would be able to access, say, the file INDEX.HTM by issuing a URL like http://some_web/info/index.htm.

It is good practice to assign unique aliases for each individual virtual server. Among other things, this will allow you to organize, for example, two access areas on one and the same server. With such an arrangement, the user accessing the server from an outside world by its symbolic Internet address (such as www.demo.ru, for example) could be taken to one home directory, but to a different directory if he/she accesses the server from within the local network using the name assigned to the host computer in that network (such as WebServ, for example). Thus, you can actually create two, three or more virtual servers (public and corporate) on one and the same machine, and arrange so (with the help of appropriate settings) that outside clients would have no access to the corporate data.

If you specify no aliases for the virtual server, only the home directory (matching the value of the *HomeDirectory* parameter of the BAIKONUR.INI file's [Baikonur] section) will exist for it.

The following example illustrates how two virtual servers that use different sets of aliases can be created.

First, we define two virtual servers in the [Baikonur.Alias] section of the BAIKONUR.INI file:

```
[Baikonur.Alias]
. . .
AliasSection = Public(www.demo.ru:80),
Corporate(WebServ:80)
. . .

Then, we specify aliases for the two sections in the
file ALIAS.FIL:

[Alias.Section.Public]
/ = c:\baikonur\home
[Alias.Section.Corporate]
/ = d:\Web\CorpInfo
```

The above demo specification means that a user accessing the server by the address www.demo.ru will be taken to the home directory c:\baikonur\home, but to the directory d:\Web\CorpInfo if he/she accesses the server using the address WebServ. The aliases specified for one section of the server remain fully invisible when another section is accessed. The same applies to BAIKONUR server's default section.

Apart from the above-mentioned type of aliases there exist aliases of two more types: name aliases and resource aliases. For more information on those refer to the section describing the format of the aliases file.

Users

The users accessing BAIKONUR server (the server's clients) may receive files from and run various programs on the server. In many situations it is rather important to determine what access rights a given user can have to files and programs and what rights a program executed on user request on the server can have. The file access rights restriction capability can be implemented only when BAIKONUR functions as a Windows NT service (i.e., uses the Windows NT file system, or NTFS).

We again stress that only files from the home directory of BAIKONUR server and its aliases (and their sub-directories) are "visible" to a user. It is these "visible" files for which we can define specific users with specific access rights.

By restricting the access rights of different users to specific files we can deny free access to certain files unless the user supplies a valid name and a password.

When And How User Access Rights Are Checked

In determining the rights of a client to request certain files and launch certain application programs, BAIKONUR server makes use of the access rights of Windows NT system users. This means that whenever it receives a request from an external client, BAIKONUR server selects one of the system users to correspond to the calling client. What specific system user is selected depends on the settings of the clients file and on the name and password supplied by the external client.

In the general case, the way the rights of BAIKONUR server clients are determined depends on the operating system used and the mode in which the server is operating, and is accomplished as follows:

- when BAIKONUR is running under Windows 95, no check of user access rights is performed;
- when BAIKONUR is running under Windows NT in the debug mode, the external client and his/her programs are granted the rights of the user who had started BAIKONUR;
- when BAIKONUR is running under Windows NT as a service and the access rights check option is disabled, the client (and the programs he/she invokes for execution) are granted the rights of that service;
- when BAIKONUR is running under Windows NT as a service and the access rights verification option is enabled, the client (and the programs he/she invokes for execution) are granted the rights defined for that client in BAIKONUR's clients file (normally, the file CLIENTS.FIL).

Thus, if you want to have all its access rights restriction capabilities implemented, your BAIKONUR server must run as a service in Windows NT (NTFS) environment.

The clients file specifies the names and passwords external to the server, along with the names and passwords of Windows NT system users corresponding to them. We shall discuss the clients file's format later in this guide. For now, let us see how BAIKONUR server is functioning as a Windows NT service when the Check Rights option (see description of the *CheckRights* parameter) is enabled.

When a client makes an attempt to access the server, the following takes place.

First, the server checks the name and password, which the client has supplied in the browser's user authorization dialog form. This form is displayed if:

- anonymous entry to the server is prohibited (see description of the *AnonymousEntry* parameter);
- the client has issued the SU command to the server;
- a program on the server has requested a password.

If anonymous entry to the server is enabled, the username and password will be blank.

Next, BAIKONUR checks if there are non-blank name and password matching those supplied by the client in the clients file (CLIENTS.FIL).

- If matching username and password are found, client-initiated programs are granted the rights of the Windows NT system user corresponding to the given username and password.
- If matching username and password are not found (or are blank), the external client receives the rights of an Anonymous user, i.e., the Windows NT system user defined as Anonymous (see below).
- If the external client has logged on to the server as an Anonymous user and no system user had been defined as Anonymous, the newcomer and his/her programs receive the rights of a service.
- If no match to the external username and/or password is found and the AutoRegistration mode is disabled (see description of the *AutoRegistration* parameter), the caller will be denied entry to the server.

Rights Of Users And Their Programs

Thus, we have outlined when and how the server performs verification of user access rights.

Clients accessing BAIKONUR server can request files from and/or run programs on it. However, the clients' file access rights and the rights of the programs executed on the server on their request differ.

A client-initiated program receives the rights of the system user specified in the *Token* parameter of BAIKONUR server's clients file (CLIENTS.FIL). If the name or password supplied by the client does not match the *Token* parameter settings, the program receives the rights of a BAIKONUR service.

An external client requesting a file receives the rights of the user group specified in the *Account* parameter of BAIKONUR's clients file (CLIENTS.FIL). If the group name is specified incorrectly or the *CheckRights* parameter is set to *Off* (signifying that the rights verification option is disabled), the client receives the rights of a service.

Administrator And Anonymous

There two special kinds of users (Administrator and Anonymous) of BAIKONUR server and each of its virtual sections. You can use the server settings to specify which of the users defined in Windows NT system correspond to these users.

The client who logs on to the sever by specifying the same username and password as those assigned to the Administrator is allowed to perform some additional functions (for example, to shut down the server). If you specify a blank name for the Administrator (or a name unknown to the system) in the settings, no one will be able to log on to the server with the rights of its Administrator.

IMPORTANT. *The Administrator possesses extended rights only with respect to BAIKONUR server, but is treated as a conventional user by Windows NT.*

All clients who log on to the server without specifying a username or password or supply a username and password that are not listed in the clients file, are given the rights of an Anonymous user (see below).

The Administrator and Anonymous are made to correspond to the users defined in Windows NT system and listed in BAIKONUR's clients file. When the ADMIN utility is employed, they are simply selected from the list of registered users.

AnonymousEntry And AutoRegistration Combination

The *AnonymousEntry* and *AutoRegistration* parameters are specified for every individual section (virtual server) of BAIKONUR server. Each of these parameters can be set either to *On* (enabled) or *Off* (disabled). By using different combinations of the *AnonymousEntry* and *AutoRegistration* parameters (in that order), you can change the server's user access rights verification logic as follows (assuming that the Anonymous user is defined and exists in the system):

- *On*, *On* – the user may enter the server without supplying a name and password or by supplying a non-existent name and password, and will receive the rights of the Anonymous user (or, more precisely, the rights of the system user corresponding to it); if the user supplies a valid name and password, he/she receives the rights of the corresponding system user.
- *On*, *Off* – the user may enter the server without supplying a password, but his/her rights will then be limited to those of the Anonymous user; or the user may supply the name and password defined in the clients file, in which case he/she will receive the rights of the corresponding system user; the input of an arbitrary name/password combination is illegal.
- *Off*, *On* and *On*, *Off* – attempts to enter the server without supplying a valid name/password combination are treated as illegal; the user must supply the name and password exactly matching those specified in the clients file, in which case he/she will receive the rights of the corresponding system user

CheckRights Parameter

The fairly straightforward access rights verification logic outlined above is supplemented by yet another parameter that has a meaning for the NT file system. Referred to as *CheckRights*, this parameters determines whether or not the rights of a client to access some specific files will be checked. The file access rights are determined by the *Account* parameter assigned to the given type of clients in BAIKONUR's clients file.

Clients File

The format of the clients file will be described in detail later in this guide. At this point, we shall merely discuss its purpose from the point of view of managing user access rights.

The clients file establishes a correspondence between the external name and password and Windows NT system user. In the general case, a client who enters the server by supplying an external name/password combination matching that specified in the clients file receives the same rights to access files and run programs as the corresponding system user.

In addition to that, the clients file makes it possible to identify external clients by their IP address rather than by the name/password combination alone (see description of the clients file). This means that clients attempting to enter the server by supplying identical name/password combinations but from different locations can be granted different access rights. To accomplish this, each "external client - system user" pair is assigned a unique identifier.

Cookies Mechanism

In situations when multiple clients work with applications, any Web server faces a difficult problem of distinguishing between individual clients. At first glance, a client can be non-ambiguously identified by his/her IP address. However, this is not quite so, because several clients belonging to a remote LAN may be working with Web server via a proxy server. In such a case they will all have the same IP address, and might well interfere with one another when attempting to access one and the same application.

To distinguish between such clients, BAIKONUR utilizes the Cookies mechanism, whereby BAIKONUR and the remote browser sort of "mutually agree" to add information unique for a given client to the data they are exchanging.

This mechanism can be useful and important in situations when clients are working with applications, but may cause problems in communicating with older browser versions.

If there is no special need in using it (such as when clients do not use any applications or it is sufficient to identify external clients by their IP address only), the Cookies mechanism can be disabled by setting the *Cookies* parameter to *Off*.

The *Cookies* parameter exists for every section (virtual server) of the BAIKONUR server employing the HTTP protocol.

File Types And Server's Response

Clients have the right to request files of various types from the server (using different client programs). The server's response to such requests varies according to the type of these files. Thus, the server will launch an executable (application) program if a file with the EXE extension is requested, but will send an image to the client if the latter requests a file with the extension GIF. In the general case, the way the server reacts to incoming client requests can be grouped as follows:

- send file (*SendSource*)
- deny access (*AccessDenied*)
- run an application program (*RunApplication*)
- run CGI program (*RunCGI*)
- execute a server command (*CLI*)

You can make the server respond to certain file types in a specific way by appropriately changing the settings of the [Baikonur.Action] section of the BAIKONUR.INI file. The format of the [Baikonur.Action] section's parameters will be discussed later, in the section describing the structure of the BAIKONUR.INI file.

IMPORTANT. *You cannot use the ADMIN utility supplied with this version of BAIKONUR to alter the parameters of the [Baikonur.Action] section.*

The type of a file is determined based on its extension. You can define the server's response to a file request for virtual servers and for individual aliases. The server uses a specific-to-general algorithm to check file types. Thus, if the BAIKONUR.INI file contains a record like

```
[Baikonur.Action]
ini = , AccessDenied
exe = text/html, RunApplication
exe/cgi-bin = text/html, RunCGI
```

the server will send an "Access Denied" message to the client if the latter requests any INI file, will run an executable program if an EXE file is requested, and will run a program in the CGI mode when the client requests an EXE file via the CGI-BIN alias.

Let us consider another example of specifying the server's response to client requests. Normally, the server is required to run an appropriate program when it receives a HTTP request for an EXE file, but should send that same EXE file to the client (rather than run the program) when it is requested using the FTP protocol. Therefore a separate virtual server (named *FTP_Serv*, for example) is usually created to handle FTP requests. The server's response to different-protocol EXE-file requests can then be specified as follows:

```
[Baikonur.Action]
exe = text/html, RunApplication
exe.FTP_Serv =, SendSource
```

There are two approaches to the use of the *AccessDenied* feature:

- *"Anything that is not allowed is prohibited"*. This approach requires that you explicitly specify what files (and from where) a client is allowed to receive. The BAIKONUR.INI file will then look something like this:

```
[Baikonur.Action]
* = , AccessDenied
* ftp = , SendSource
exe = text/html, RunApplication
exe/cgi-bin = text/html, RunCGI
htm = text/html, SendSource
html = text/html, SendSource
```

```
txt = text/plain, SendSource
gif = image/gif, SendSource
jpg = image/jpeg, SendSource
jpeg = image/jpeg, SendSource
xls = binary/excel, SendSource
. . .
```

- *"Anything that is not prohibited is allowed"*. This approach requires that you explicitly specify what files a client is not allowed to receive. The BAIKONUR.INI file will then look approximately as follows:

  ```
  [Baikonur.Action]
  * = , SendSource
  exe = text/html, RunApplication
  exe/cgi-bin = text/html, RunCGI
  ini = , AccessDenied
  fil = , AccessDenied
  ```

* These parameters will be described in greater detail later, in the section devoted to the format of BAIKONUR's system files.

The format of the [Baikonur.Action] section allows you to do some tricks by arranging so that when a request for a file of some specific type is received the server would run an application which can itself decide what (if anything) should be sent to the client. Specifically, BAIKONUR's ISAPI support capability is based on the use of this trick: when it receives a request for a DLL file, the server runs a special program that loads the ISAPI DLL, accesses its functions and returns the appropriate response. Here is an example of how this is achieved:

```
[Baikonur.Action]
dll = , RunApplication, isapiter.exe, **?
```

The above declaration means that when a DLL file request is received (for instance, like http://some_web/counter.dll?id=601&name=User_12), the server will run the ISAPITER.EXE program using a command line in which the name of and path to the requested DLL file and the request parameters will be specified (the string following the question mark "?" in the URL).

Tasks And Settings Examples

It might be difficult for you to immediately grasp BAIKONUR server's extensive administration capabilities, especially as far as the management of client access rights is concerned. To make things easier for you, let us consider some practical examples. More detailed recommendations on how a BAIKONUR Web site can be created and how its security can be ensured will be discussed later in this guide.

Default Settings

BAIKONUR is ready to perform its functions as a Web site server immediately after installation. However, it is still desirable to make some changes to BAIKONUR's default settings before you practically use it.

After installation, the server supports the HTTP, HTTPS, FTP, Gopher and Finger protocols; the sections for the HTTP, FTP and Gopher protocols (and their home directories) are created; the Administrator and Anonymous users for each of these sections are defined (although the clients corresponding to them may not yet be defined in your system); the clients have access to BAIKONUR demo applications and several CGI and ISAPI examples; and the Yandex system is installed (in Russizn version), although the documents are not indexed yet.

We recommend that you make the following changes to BAIKONUR server's standard operating mode (default) settings:

- replace the server's home page and remove the demo programs;
- remove the CGI and ISAPI program examples;
- disable all protocols you do not intend to use (for instance, HTTPS and Gopher);
- create a user in Windows NT and make it correspond to Anonymous;
- index documents for use with the Yandex system (in Russian version).

A Public Server

Suppose you want to build a Web site that will contain public-accessible information. Access to the site is to be by the HTTP or FTP protocol. No other protocols is planned to be used. The site may contain applications operating with SQL server databases.

To achieve this, you must do the following:

- Disable (remove) all unused protocols, leaving only the HTTP (port 80) and FTP (port 21).
- Create a Windows NT user (say, with the name WEB_USER and the password WEB_PASS), place this user in the *Users* group, and check if that user will have the right to launch database applications.
- Create the following record in the clients file (by editing it directly or using the ADMIN utility for the purpose):

```
[Guest_ID]
    Name =
    Password =
    Token = WEB_USER:WEB_PASS
    Account = Users
```

- Change the [Baikonur.Client] section's default settings as follows:

```
[Baikonur.Client]
    AnonymousEntry = On
    AutoRegistration = On
    CheckRights = On
    Cookies = On
    Anonymous = *Guest_ID
```

A Corporate Server

Suppose now that you wish to create a Web site, which no clients can access unless they supply a certain valid username/password combination. Only the HTTP protocol is planned to be used to address the site.
To achieve this, follow these steps:

- Disable (remove) all unused protocols, leaving only the HTTP (port 80).
- Create a Windows NT user (say, with the name WEB_USER and the password WEB_PASS), place this user in the *Users* group, and check if that user will have the right to launch database applications.
- Create *N* records in the clients file for every registered client as illustrated below (you can do it with the aid of the ADMIN utility):

```
[Smith.ID]
    Name = Smith
    Password = Djkrjd
    Token = WEB_USER:WEB_PASS
    Account = Users

[Falkon.ID]
    Name = Falkon
    Password = Qwert
    Token = WEB_USER:WEB_PASS
    Account = Users
```

- Change the [Baikonur.Client] section's default settings as follows:

```
[Baikonur.Client]
    AnonymousEntry = Off
    CheckRights = On
    Cookies = On
```

With these settings, the client will be queried for a username/password combination when he/she first tries to log on to the server, and will be denied access to the server information if a valid combination is not supplied.

A Combination (Public/Corporate) Server

This case is essentially a combination of the former two, whereby one and the same a Web site contains both public and corporate information (and applications). This situation calls for the use of sections (virtual servers). With this approach, you can arrange so that public information would be accessible through one port (say, port 80), while corporate information would be accessible through another port (say, port 8000).

All you basically need do to achieve this is to create a section corresponding to port 8000 and deny access to it without a password, and leave port 80 as the default one.
To do so, proceed as follows:

- Perform all the steps as for a public server.
- Add users like you did it for a corporate server.
- Create a virtual server for port 8000:

```
[Baikonur.Alias]
AliasSection = CorpServ(:8000)
. . .
[Baikonur.Client]
section.CorpServ=, , Basic, Off, Off, On, On
```

- Specify aliases for the new section in the ALIAS.FIL file, for example like this:

```
; corporate server's home directory
[Alias.Section.Corp.Serv]
/ = d:\corporate\info
```

To enter your corporate server, the user will now have to specify (or click on the link to) a URL of the form http://www.some_web.ru:8000/, and supply his/her name and password.

Joint Use Of BAIKONUR And Other Web Servers

It may turn out that some Web server other than BAIKONUR (such as Microsoft's IIS, for example) had already been chosen as your company's standard Web server. Even so, BAIKONUR can still be successfully employed as an application server. You simply assign the HTTP protocol to an unused port, and it is this port that all clients wishing to work with applications will be accessing BAIKONUR through.
If your company has had some experience in developing ISAPI logic for a MS IIS server, BAIKONUR is again the most sensible choice for handling ISAPI tasks, because the techniques it utilizes to work with ISAPI DLLs is safer from the point of view of Web server security and stability.

Server's Command Language

BAIKONUR server's command language is a set of standard pre-defined *lexemes,* which the server interprets as commands to execute certain procedures.
In the HTTP protocol, the server commands are issued as part of the URL, for example, like http://some_web/as.
Listed below are the commands accessible to all clients of the server.

• **as** -	command to output a list of aliases on the server;
• **ts** -	command to output a list of active tasks in the client's current session (in HTML standard);
• **application.!** -	command to terminate the application;

- **!** - command to terminate all applications;
- **application.!?terminate** - command to terminate the application (terminate process);
- **!?terminate** - command of forcibly terminate all applications;
- **su** - command to change the name of the server's client;
- ***exit** - command to end the client's current session;
- **sl** - command to output a list of sessions with identical usernames but different IP addresses (this command is accessible only to the clients registered in Windows, and is inaccessible to Anonymous users);
- **echo** - command to echo the input information.

The following commands are accessible only to the server's Administrator:

- ***cs** - command to output a list of the server's clients;
- ***addalias** - command to add a new alias;
- ***edalias** - command to edit the alias;
- ***addclient** - command to register a new client;
- **shutdown** - command to shut down the server;

(* - not implemented in current version).

Formats Of BAIKONUR's System Files

BAIKONUR server employs the following system files:

- initialization file (BAIKONUR.INI);
- aliases file (normally, ALIAS.FIL);
- clients file (normally, CLIENTS.FIL);
- SSL protocol certificates file (normally, CRTFCT.FIL).

The formats of these files are described below.

BAIKONUR Server Initialization File

The first thing that a BAIKONUR server does after it is started up is to read its initialization file, where values of the server's various parameters are stored. The initialization file is located in Windows home directory and is named BAIKONUR.INI.

Purpose Of Initialization File

The following parameters of BAIKONUR server can be defined in its initialization file:

- the server's system directory, i.e., the directory in which various service components of the server (such as protocol DLLs) are located;

- level of the server's diagnostics (system-generated messages can be output to a file or to the console.).

You can also use the initialization file to:

- determine the amount of diagnostic information to be output by your BAIKONUR server;
- assign a different name to the server's log file and specify its type;
- determine the timing parameters of connects and the degree of multisequencing of requests;
- determine the server's home directory and its home page;
- determine the file describing the server's virtual directories;
- re-assign names and passwords of the server's Administrator and Anonymous users;
- determine the types of files that can be processed by the server, and specify the way these files are to be processed.

Structure Of Initialization File

BAIKONUR server's initialization file is a conventional Windows initialization file. Normally, it consists of the following five sections:

[Baikonur] - basic parameters of the server;
[Baikonur.Protocol] - settings of the server's ports;
[Baikonur.Alias] - settings for the system of virtual directories and re-assignments (aliases);
[Baikonur.Action] - settings for the processed file types and handlers;
[Baikonur.Client] - settings for the access rights management system.

The [Baikonur] section must always be the first one in the initialization file.

Parameters of [Baikonur] Section

The [Baikonur] section of the initialization file is used to define the following parameters of the server:

1. *LogLevel* parameter
   determines the level of warning messages to be output to the log file;
2. *LogFile* parameter
   determines the name and type the log file;
3. *LogDIrectWrite* parameter
   determines the method to be used to output system messages to the log file;
4. *SystemDirectory* parameter
   declares the server's system directory;
5. *HomeDirectory* parameter
   declares the server's home directory;
6. *HomePage* parameter
   declares the server's home page;
7. *DirInfo* parameter

specifies the name of the file with information on the FTP directory;

8. *Agents* parameter
determines the number of pending request handlers;
9. *Managers* parameter
determines the number of pending I/O handlers;
10. *TimeResolution* parameter
determines the time interval between successive server status checks;
11. *TimeClientInactive* parameter
determines the time interval during which an open but inactive connect to a client may exist;
12. *TimeClientDisactive* parameter
determines the time interval during which the server is to keep trying to send data to the client;
13. *MaxClientConnections* parameter
determines the maximum number of client connections that may be concurrently opened;
14. *MaxClientApplications* parameter
determines the maximum number of active applications per client;
15. *Applications* parameter
determines the maximum number of applications that can be concurrently active on the server;
16. *ApplicationTimeOut* parameter
determines the time interval after the expiration of which the active application is to receive a “timeout” signal;
17. *ApplicationStartTimeOut* parameter
determines the time interval during which the server must wait for a response from the application being started and should not terminate its process;
18. *PassiveDisable* parameter
determines the method to be used to open a data transfer channel for an FTP client;
19. *Station* parameter
declares the name of the server’s virtual workstation;
20. *DeskTop* parameter
declares the name of the server’s virtual desktop manager;
21. *SocketNoDelay* parameter
determines the data send delay mode to be used to generate a full IP package;
22. *SocketRcvBuf* parameter
determines the size of the system’s IP package receive buffer;
23. *SocketSndBuf* parameter *f*
determines the size of the system’s IP package transmit buffer;
24. *SessionTimeOut* parameter *t*
determines the time interval after the expiration of which the non-active client’s session is to be terminated;
25. *ClientsFile* parameter
declares the name of the file describing the system clients;
26. *CertificateFile* parameter

determines the name of the file containing the server's SSL protocol certificates;

27. *MaxReceiveBuffer* parameter
    determines the maximum size of the first block of received data.

LogLevel Parameter

The *LogLevel* parameter determines the level of warning messages to be output to BAIKONUR's log file. This parameter must always precede the log file's name and type declaration (the *LogFile* parameter).
The following warning message levels are distinguished:

1, *FatalError*
fatal error messages; these are displayed onscreen in the form of a STOP dialog box regardless of the log file settings; a fatal error results in termination of the server.

2, *CriticalError*
critical error messages; a critical error is one that *may* make the system inoperable.

3, *Error*
other error warning messages.

4, *OK*
message signifying that an operation has been successfully executed.

5, *Warning*
error or access rights violation message.

6, *Message*
messages output to the server's log file by clients.

7, *Application*
messages output to the server's log file by applications.

8, *Debug*
debug messages.

9, *Debug1*
design-time debug messages.

10, *ALL*
log all parameters.

All valid options in a parameter line should be comma-separated. If the *ALL* value is specified and is followed by a list of messages, the latter will be not be output.
Example Syntax:

```
;output messages OK, FatalError and Message:
LogLevel = OK, FatalError, Message

;output all messages except OK and Message:
LogLevel = ALL, OK, message
```

LogFile *Parameter*

The *LogFile* parameter determines the type and name of the system log file. Two logging methods are supported:

1. *FILE* – output to a hard disk file;
2. *CON*[*SOLE*] – output to the system console.

The log file's type and name should be comma-separated. The name of the file specified for *CONSOLE*-type output is used as the console window's header. If no *LogFile* parameter is specified, only messages of the *FatalError* level are output to the system console.
Example Syntax:

```
;do not create a log file
LogLevel =

;create log as a desktop console with "Baikonur" header
LogLevel = CON, Baikonur

;create log as a file named "Report.txt"
;in server's default directory
LogLevel = FILE, Report.txt
```

NOTE. The CONSOLE file type should be used only when BAIKONUR server is started up in the debug mode (with the switch "-d" in the command line) or when it is started up by a user manually as a service and the service system option is set so as to enable that service to communicate with the console.

LogDirectWrite *Parameter*

The *LogDirectWrite* parameter determines the method to be used to output system messages to the log file. The default method is parallel output of messages to the log file with the help of request handlers. The default value of this parameter is *Off*.
Example Syntax:

```
LogDirectWrite = On
```

SystemDirectory *Parameter*

The *SystemDirectory* parameter determines the directory containing BAIKONUR server's service components, such as libraries of the server administrator and libraries of the server-supported protocols. Once this parameter is declared, the directory defined in it becomes BAIKONUR's current system directory, with the result that the latter becomes the "default" directory for all subsequent file declarations in the server's INI file.
Example Syntax:

```
SystemDirectory = d:\Baikonur\Sys
```

The above declaration means that all the file names mentioned in the INI file will be searched for in the directory Baikonur\Sys on disk d:

HomeDirectory *Parameter*

The *HomeDirectory* parameter determines the server's home directory, i.e., the directory which the resource files will be read from in situations when no directory is specified in the URL.
Example Syntax:

```
HomeDirectory = d:\Baikonur\HOME
```

The above declaration means that when a URL like http://www.baikonur/advert.htm is received, the file *advert.htm* will be searched for in the directory d:\Baikonur\HOME.

HomePage *Parameter*

The *HomePage* parameter determines the name of the file to be loaded by default, i.e., the name of the file which the server will address in situations when the client tries to access a directory or an alias without explicitly specifying the name of the requested resource.

Example Syntax:

```
HomePage = Home.htm
```

The above declaration means that when the server receives a URL like http://www.baikonur.com/, the latter will be parsed as http://www.baikonur.com/Home.htm, and the client will see the page Home.htm.

DirInfo *Parameter*

The *DirInfo* parameter determines the name of the file containing information on FTP directories. When a directory is viewed, this file will be loaded by default after the directory's header.
The default value of the parameter is *dirinfo*.
Example Syntax:

```
DirInfo = readme.txt
```

Agents *Parameter*

The *Agents* parameter determines the number of pre-launched active threads (referred to as *IP agents* hereinafter) designed to service the tasks related to the handling of client requests and requests to output information to the log file.
The minimum number of *IP agents* is 2; the maximum number is 64.
Example Syntax:

```
Agents = 16
```

Managers *Parameter*

The *Managers* parameter determines the number of pre-launched active threads

(referred to as *IP managers* hereinafter) designed to service the I/O operations through currently open connects to clients.
The minimum value of the parameter is 2; the maximum value is 64.
Example Syntax:

```
Managers = 16
```

TimeResolution *Parameter*

The *TimeResolution* parameter determines the time rate at which to run the server status monitoring procedure (designed to check how long the open connects are being used, how long the clients are working with the server, as well as the server's other time-related parameters).
The value of this parameter is specified in milliseconds. The minimum value is limited to 20 milliseconds; the maximum value is unlimited.
Example Syntax:

```
TimeResolution = 5000
```

TimeClientInactive *Parameter*

The *TimeClientInactive* parameter determines the time interval during which the connect to a client may exist pending the arrival of a request, or how long the connect may be kept alive after a request is processed (i.e., till the arrival of a new request in situations when the connect is of the *Keep-Alive* type).
The value of this parameter is specified in seconds.
Example Syntax:

```
TimeClientInactive = 120
```

TimeClientDisactive *Parameter*

The *TimeClientDisactive* parameter determines the time interval during which the server is to keep trying to send data to the client with whom it has a normal connect. *TimeClientDisactive* is counted from the instant of the most recent successive attempt to send a portion of data to the client.
The value of this parameter is specified in seconds.
Example Syntax:

```
TimeClientDisactive = 120
```

MaxClientConnections *Parameter*

The *MaxClientConnections* parameter determines the maximum number of open connections to one client that may concurrently exist on the server.
Example Syntax:

```
MaxClientConnections = 5
```

MaxClientApplications *Parameter*

The *MaxClientApplications* parameter determines the maximum number of active applications per client. Shared applications are counted on a per-client basis.
Example Syntax:

```
MaxClientApplications = 15
```

Applications *Parameter*

The *Applications* parameter determines the maximum number of applications that can be concurrently active on the server. Shared applications are counted as a single application for all the clients.
Example Syntax:

```
Applications = 60
```

SessionTimeOut *Parameter*

The *SessionTimeOut* parameter determines the time interval that must expire before the client session that has no active connects or running applications shall be terminated by the server.
SessionTimeOut is counted from the instant of the most recent activity of the client with respect to the server. The value of this parameter is specified in seconds. The minimum value is 60; the maximum value is 3600.
Example Syntax:

```
SessionTimeOut = 300
```

ApplicationTimeOut *Parameter*

The *ApplicationTimeOut* parameter determines the time interval after the expiration of which the active application is to receive an “application timeout” signal.
ApplicationTimeOut is counted from the instant of the most recent activity of the application with respect to the server, and is specified in seconds.
Example Syntax:

```
ApplicationTimeOut = 120
```

ApplicationStartTimeOut *Parameter*

The *ApplicationStartTimeOut* parameter determines the time interval during which the server must wait for a response from the application being started and should not terminate the application process if the client issues an application_name.!?terminate command.
The value of this parameter is specified in seconds.

Example Syntax:

```
ApplicationStartTimeOut = 120
```

PassiveDisable ***Parameter***

The *PassiveDisable* parameter determines the method to be used to open a data transfer channel for an FTP client (i.e., it enables or disables the server's FTP service passive mode). By default, this parameter is turned off (implying that the service passive mode is enabled).
Example Syntax:

```
PassiveDisable = Off
```

Station ***Parameter***

The *Station* parameter determines the name of the server's virtual workstation (i.e., the name of the station on which the server manager window will be opened to execute applications in the service mode).
The default name of the workstation is BaikSta0.
Example Syntax:

```
Station = BaikSta0
```

Desktop ***Parameter***

The *Desktop* parameter determines the name of the server's virtual desktop manager to execute applications in the service mode.
The default name of the desktop manager is Workshop.
Example Syntax:

```
Desktop = Workshop
```

SocketNoDelay ***Parameter***

The *SocketNoDelay* parameter determines whether or not the data send delay mode is to be used to generate a full IP package using the Nagle algorithm. The default value of this parameter is *On (*implying that the Nagle algorithm is not used).
Example Syntax:

```
SocketNoDelay = On
```

SocketRcvBuffer ***Parameter***

The *SocketRcvBuffer* parameter determines the size of the system's IP package receive buffer. The default value of this parameter is 8192.
Example Syntax:

```
SocketRcvBuffer = 8192
```

SocketSndBuffer *Parameter*

The *SocketSndBuffer* parameter determines the size of the system's IP package send buffer. The default value of this parameter is 8192.
Example Syntax:

```
SocketSndBuffer = 8192
```

ClientsFile *Parameter*

The *ClientsFile* parameter declares the name of the file containing the coded names of the server's clients, paths to the their home directories, and information needed to keep a session log. The clients file is created and updated by the server every time a client is registered on the server.

Example Syntax:

```
ClientsFile = c:\winnt40\system\userscfg.fil
```

CertificateFile *Parameter*

The *CertificateFile* parameter determines the name of the file containing the server's SSL certificates.
Example Syntax:

```
CertificateFile = c:\winnt40\system\crtfct.fil
```

Parameters of [Baikonur.Protocol] Section

The [Baikonur.Protocol] section of the initialization file is used to declare the ports, libraries and parameters of the protocols used by the server.
The general format of the declaration is as follows (optional parameters are enclosed in square brackets]:

```
protocol-specifying_part_of_URL:port =
protocol_name_and_version, dll_name,
protocol_parser_name,
[port_activity_at_startup], [security_code],
[security_scheme], [client_identification_scheme],
[administration_capability]
```

In this declaration:
protocol-specifying_part_of_URL
is the abbreviation defining the name of the protocol (for example: http, ftp);
port
is the decimal value of the port assigned for the protocol (for example: 80 for http, or 21 for ftp);

protocol_name_and_version
is the name and version of the protocol (for example: HTTP/1.0);
dll_name
is the name of the DLL file containing the protocol parser. If the path to it is omitted, the file is searched for in the directory specified in the *SystemDirectory* parameter;
[port_activity_at_startup]
determines whether or not the port is to be listened to at startup; valid value is *On* or *Off*; the default value is *On*;
[security_scheme]
is the port access security scheme at transport level (for example: SSL);
[client_identification_scheme]
is the scheme employed to identify the client by the port at protocol level (for example: Basic); the default value is *Basic*;
[administration_capability]
determines whether or not administration of the server can be performed through the given port; valid value is *On* or *Off*; the default value is *On*;

Example Syntax:

```
http:80=HTTP/1.0,HTTP10.DLL,_Parser

http:8080=HTTP/1.0,HTTP10.DLL,_Parser,Yes,,Basic,Yes

https:443=HTTPS/1.0,HTTP10.DLL,_Parser,Yes,SSL,Basic,Yes

ftp:23=FTP/1.0,ftp.dll,_FTPParser
```

Here is an example of how a restricted-access SSL port can be organized:

```
[Baikonur]
.
.
CertificateFile = c:\winnt\system\security\server.pem
.
.

[Baikonur.Protocol]
.
https:443=HTTPS/1.0,HTTP10.DLL,_Parser,Yes,SSL,Basic,Yes
.
```

NOTE. You can use different ports (or even several ports per protocol) and combinations thereof with libraries and protocol parser functions.
The need to carefully describe the supported protocol(s) in the server's initialization file stems from the fact that BAIKONUR Web App Server has a special feature known as "dynamic protocol switching" capability. This capability makes it possible to connect similar-level file transfer protocols (such as http, ftp, smtp, nmtp, etc.) to a BAIKONUR server.

To have a new protocol connected to the server, it is essential to correctly specify the corresponding parameters of that protocol and its parser function. When the server receives a URL request, the part of the URL string following the protocol name (acronym) goes to the parser function for further parsing.
The procedures employed to develop new protocols and connect them to the system are described in the documentation supplied with senior versions of BAIKONUR. The version under discussion merely allows you to connect any protocols developed with the aid of the tools available in BAIKONUR's senior versions.

Parameters of [Baikonur.Alias] Section

The [Baikonur.Alias] section of the initialization file is where the server's directory system parameters and substitute file names (aliases) are specified.

AliasTranslation *Parameter*

The *AliasTranslation* parameter is used to enable or disable the alias translation feature. By default, this feature is enabled (*On*).
Example Syntax:

```
AliasTranslation = On
```

AliasFile *Parameter*

The *AliasFile* parameter declares the name of the file describing the server's alias system. By default, this file is searched for in BAIKONUR's system directory. The format of the aliases file will be discussed later in the text.

Example Syntax:

```
AliasFile = alias.fil
```

AliasSections *Parameter*
(AliasSection, AliasSection0-AliasSection9, AliasSectionA-AliasSectionF)[*]

The *AliasSections* parameter declares the names of the systems of aliases for the server's sections (otherwise called *virtual servers*).
A section name is a string of 0 to 16 characters containing no blanks or commas. Each name declared in a section belongs to that section and its corresponding list of aliases in the *AliasFile* file. Each section can be linked to a list of comma-separated "*server_name:server_port*" declarations. If the port is not specified, the section for the given server name will be associated with any port. If the server name is omitted, then the section will be associated with the given port.
The algorithm employed to search for and determine the necessary section is as follows.

```
if server name is declared, then

{
        search for section by server name and valid port
        if the section is not found, then
        search for section by server name and default port
    }

    if section is not found, then
        search for section without server name and valid
    port
    if section is not found, then
        search for section without server name and default
    port
    if section is not found, then
        return default section

    return thus found section
```

The general syntax of the *AliasSections* parameter is as follows:

```
AliasSections = [section_name]
([server_name]:[port_name]
{[, . . . : . . .]} {[, . . .]}
```

Example Syntax:

```
AliasSection = HTTP_Server (www.baikonur.demo.ru :
80, www.demo.ru : 80) , HTTP_CorporateServer
(baikonur : 80 , localhost : 80) FTP_Server (: 21),
anyserver (anyserver:, Default_Server (:)
```

NOTE. The section name should also be taken into account in file type declarations (see below).

[Baikonur.Action] Section

The [Baikonur.Action] section is where we declare the processed file types, specify handlers to process files of these types, and set up their parameters.

File Type Declarations In [Baikonur.Action] Section

The type of a file and the method to be employed to process it while servicing a request are determined by the server based on the file's extension specified in the request, or the results of interpretation of the appropriate alias.
There exist the following file action identifiers which the server is capable of interpreting:

AccessDenied
deny access to resource;
SendSource
send out static resource;
RunApplication
run application using the BAIKONUR server interface;

RunCGI
run application using the CGI interface;
CLI
use file extension for generating a server command.
The general format of a file type declaration is as follows (optional parameters are enclosed in square brackets):

```
file_extension [. section_name][/alias] = MIME-type,
file_action_identifiers [, handler_program_name]
[,handler_program_switches]
```

In this declaration:
file_extension
string denoting the file name's extension (for example: txt or html); the extension string should contain no dots, blank or special characters; blank characters, if any, are replaced with the percent character ("%");
section_name
is the name of the server's section within which this declaration is valid;
alias
is the auxiliary identifier string (protocol-dependent); it is interpreted as the name of the alias for the HTTP protocol;
MIME-type
denotes the file's MIME-type (for example: text/plain or text/html);
file_action_identifiers
identify the actions to be performed with the file; these identifiers are taken from the list of valid identifiers (for example: SendSource or RunApplication);
handler_program_name
is the name of the handler program, i.e., the program that must be activated on the server when a file of the given type is accessed (for example: myhttp.exe);
handler_program_switches
are the auxiliary switches for the handler program, i.e., a program execution mode string to be appended to the handler program's command line as its first parameter every time a request for the given resource is received.

Remarks On File Extension Declarations

A file extension in the form of a single asterisk ("*") is used to denote (a)_the action which the server must perform when the client attempts to address a file of an undeclared type, or (b)_the MIME-type, which the server should automatically assign to the file if its extension is unknown. The inclusion of a handler name and its switches makes the server run the program designed to handle unknown extensions. When the extension ("*") is declared for one of its sections, the server limits interpretation of the extensions to within the given section.

Remarks On Handler Program Declarations

You can use some special characters to declare handler programs, including:

* - denotes a file name without extension; when this character is included in the declaration, the name of the file (without extension) and its directory path are passed to the handler program’s command line;

** - denotes a file name with extension; when this character is included in the declaration, the name of the file complete with its extension and directory path are passed to the handler program’s command line;

? - denotes a parameter; when this character is included in the declaration, the parameter specified in the request (i.e., the string following the “?” character in the URL) is passed to the handler program’s command line.

For example, the declaration wap=text/plain, runapplication, *.exe will make the server run the program program.exe when an attempt is made to access the resource program.wap.

And another example. If we make the declaration app=text/plain, runapplication, processit.exe, *, a call to the resource Xproc.app will make the server run the program processit.exe” with path/Xproc as the command line’s first parameter.

Notes On Use of Directory Paths In Handler Name Declarations

- If no directory path is specified in the handler name, the server will run the program using the directory (or its alias) specified in the resource locator.
- Aliases are interpreted by the server before it interprets the handler programs.

Example Syntax:

```
;the following declaration will make the server
;run BAIKONUR applications
exe = , RunApplication

;with the following declaration, the server
;will run CGI applications
;when request is made from CGI-BIN alias
exe/cgi-bin = , RunCGI

;the following declaration will make the server
;post an “Access Denied” message to the client
;whenever he/she tries to access an .INI file
ini = , AccessDenied

;with the following declaration, the server
;will be allowed to send .INI files to clients
ini = text/plain, SendSource
```

Parameters Of [Baikonur.Client] Section

The [Baikonur.Client] section serves to declare the basic parameters employed to authenticate clients and verify their access rights.

AuthenticationScheme *Parameter*

The *AuthenticationScheme* parameter determines the default client authentication scheme.

Example Syntax:

```
AuthenticationScheme = Basic
```

NOTE. In its current version, BAIKONUR supports only one client authentication scheme known as ***Basic***. The *AuthenticationScheme* parameter has been introduced to permit future add-ons.

AnonymousEntry *Parameter*

The *AnonymousEntry* parameter is employed to enable or disable the server's anonymous client entry mode.
When this mode is enabled, any anonymous client will be able to request and receive information from the server without supplying a username and a password. With this arrangement, the system will treat every new client as having the name defined in the *Anonymous* parameter (see below).
If the *AnonymousEntry* mode is disabled, the server will query every new client for a valid name/password combination before granting him/her access to the server's resources.
The default value of the *AnonymousEntry* parameter is *On (*enabled*)*.

Example Syntax:

```
;anonymous entry is enabled (default setting)
AnonymousEntry = On

;the following declaration disables anonymous entry
AnonymousEntry = Off
```

AutoRegistration *Parameter*

The *AutoRegistration* parameter is employed to enable or disable automatic registration of new clients on the server.
When the *AutoRegistration* mode is enabled, any anonymous client will be allowed to register on the server with the access rights and under the name declared in the *Anonymous* parameter.
The name supplied by the client in the browser's user authentication dialog will actually be treated as the name of a separate session of an Anonymous client.

If the *AutoRegistration* mode is disabled, only the users registered in Windows NT system will be allowed access to the server's resources.
The default value of the *AutoRegistration* parameter is *On (*enabled*)*.

Example Syntax:

```
;automatic registration is enabled (default setting)
AutoRegistration = On

;automatic registration is disabled
AutoRegistration = Off
```

CheckRights *Parameter*

Depending on its setting, the *CheckRights* parameter enables or disables verification of the rights of users to access files at the operating system level. By default, the *CheckRights* mode is enabled when the server runs under Windows NT, is always disabled when the server runs under Windows 95, and is also disabled when the server is used in the debug mode.

Example Syntax:

```
;user access rights verification is enabled
CheckRights = On

;user access rights verification is disabled
CheckRights = Off
```

Administrator *Parameter*

The *Administrator* parameter declares the name and password of the server administrator. Note that the administrator name and password are never explicitly specified in the server's initialization file. Instead, the latter contains a text string in which both the name and the password are encrypted. In situations when the server is required to support several user authentication schemes, a string encrypted with the use of a different algorithm may be assigned to the *Administrator* parameter.
There are two basic *Administrator* declaration formats:

1. Explicit declaration of the administrator's name and password encrypted with the use of the Base64 algorithm:

Administrator = <Basic-Cookie>

2. Declaration of the administrator's name and password via reference to the client's ID in the *ClientsFile* parameter:

Administrator = *<client_ID>
where client_ID is the ID of the client or section in the clients file.
The default name and password of the administrator are "Administrator" and "Baikonur", respectively.
To alter the administrator's name and password encrypted with the use of the Base64

algorithm, use the “Make COOKIE” utility (MkCookie.exe) that comes with BAIKONUR, and substitute the resulting value for <Basic-Cookie> in the *Administrator* parameter.

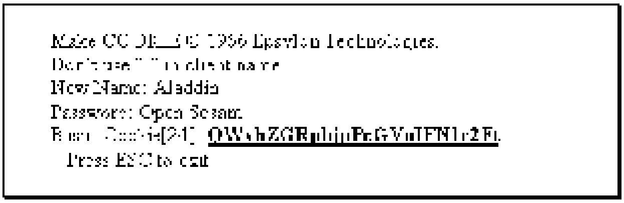
NOTE. DO NOT use colon (“:”) in client names. REMEMBER that client names and passwords are CASE-SENSITIVE.

Example Syntax:

```
Administrator = QWxhZGRpbjpPcGVuIFN1c2Ft

Administrator = *ADMIN_ID
```

The first of the above *Administrator* parameter values has been obtained by copying the string of characters from the “Basic-Cookie[24]” line of the MkCookie.exe utility (ref. Figure 1); in this case, the name of the server’s administrator has been changed to “*Aladdin*” and assigned the password “*Open Sesam*”.



```
Make COOKIE (C) 1996 Epsylon Technologies.
Don't use ":" in client name
New Name: Aladdin
Password: Open Sesam
Basic-Cookie[24]: QWxhZGRpbjpPcGVuIFN1c2Ft
  Press ESC to exit
```

Figure 1. MkCookie.exe Utility’s Sample Dialog

Anonymous *Parameter*

The *Anonymous* parameter declares the default name and password of the server’s client (anonymous user).

There are two basic formats of the *Anonymous* declaration:

1. Explicit declaration of the anonymous client’s name and password encrypted with the use of the Base64 algorithm:

Anonymous = <Basic-Cookie>

2. Declaration of the anonymous client’s name and password via reference to the client’s ID in the *ClientsFile* parameter:

Anonymous = *<client_ID>

where client_ID is the ID of the client or section in the clients file.

By default, anonymous clients are assigned the name “Anonymous” and ***no*** password.

To alter the anonymous client’s name encrypted with the use of the Base64 algorithm, use the “Make COOKIE” utility (MkCookie.exe) that comes with BAIKONUR, and substitute the value it returns for <Basic-Cookie> in the *Anonymous* parameter.

NOTE. DO NOT use colon (“:”) in client names. REMEMBER that client names and passwords are CASE-SENSITIVE.

Example Syntax:

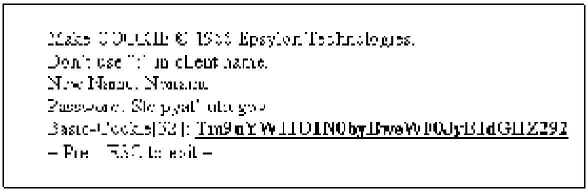
---

* Additional section names were introduced because lines in the alias file have but a limited length.

```
Anonymous = Tm9uYW11O1N0byBweWF0JyB1dGl1Z292

Anonymous = *GUEST_ID
```

The value of *Anonymous* parameter can be obtained by copying the string of characters from the “Basic-Cookie[32]” line of the MkCookie.exe utility, as shown in the sample dialog in Figure 2. Here, the name of the server’s anonymous client has been changed to “*Noname*” and assigned the password “*Sto pyat’ utiugov*”. Once declared, this username/password combination will verified every time a new client first accesses the server from his/her browser.



```
Make COOKIE © [illegible] Epsylon Technologies.
Don't use ':' in client name.
New Name: Noname
Password: Sto pyat' utiugov
Basic-Cookie[32]: Tm9uYW11O1N0byBweWF0JyB1dGl1Z292
– Press ESC to exit –
```

Figure 2. MkCookie.exe Utility’s Sample Dialog

Section *Parameter*

The *Section* parameter is used to specify the server’s settings for a specific section. The general syntax for the *Section* parameter declaration is as follows:

```
Section.section_name = administrator,
anonymous_client_name_64, scheme,
anonym_entry, autoregistr, check_rights
```

In this declaration:

```
section_name
```

is the name of the server’s section (similar to the *AliasSection* parameter of the [Baikonur.Alias] section);

administrator

denotes the name and password of the server administrator encrypted with the use of the Base64 algorithm, or reference to the section’s administrator declared in the *ClientsFile* parameter (similar to the *Administrator* parameter);

anonymous_client_name_64

denotes the name and password of the section’s anonymous client encrypted with the use of the Base64 algorithm, or the corresponding reference to the *ClientsFile* parameter (similar to the *Anonymous* parameter);

scheme

is the scheme employed to authenticate the server’s clients (similar to the *AuthenticationScheme* parameter);

anonym_entry

enables or disables anonymous entry of clients to this section of the server (similar to the *AnonymousEntry* parameter);

autoregistr

enables or disables automatic registration of new clients on the server (similar to the *AutoRegistration* parameter);

check_rights

enables or disables verification of clients' file access rights at the operating system level (similar to the *CheckRights* parameter);

Example 1

Declaration with the use of the Base64 encryption algorithm:

```
Section.HTTP_CorporateServer = QWxhZGRpbjpPcGVuIFN1c2Ft,
Tm9uYW11O1N0byBweWF0JyB1dGl1Z292, Basic, No, Yes, Yes
```

Example 2

Declaration with the use of references:

```
Section.HTTP_CorporateServer = *admin_id, *anonymous_id,
Basic, No, Yes, Yes
```

Example 3

Registration of anonymous clients is disabled:

```
Section.HTTP_CorporateServer = *admin_id, *anonymous_id,
Basic, No, Yes, Yes
```

Example 4

Registration of anonymous clients is disabled:

```
Section.HTTP_CorporateServer = QWxhZGRpbjpPcGVuIFN1c2Ft,
Tm9uYW11O1N0byBweWF0JyB1dGl1Z292, Basic, No, Yes, Yes
```

Example 5

No administrator is declared for the section - section administrator functions are disabled:

```
Section.HTTP_CorporateServer = ,
Tm9uYW11O1N0byBweWF0JyB1dGl1Z292, Basic, Yes, Yes, Yes
```

Example 6

Neither administrator nor anonymous client is declared for the section - section administrator functions are disabled; password is always requested; the *anonym_entry* parameter is ignored:

```
Section.HTTP_SecureServer = , , Basic, Yes, Yes, Yes
```

Aliases File

The server's aliases file makes it possible to define virtual directories for the server and assign virtual names for its resources, as well as re-assign the server's home directory and its home page.

BAIKONUR uses aliases of the following three basic types:

- directory alias;

- resource alias;
- name alias.

Directory Alias

The directory alias mechanism enables you to substitute a certain name (alias) for any directory named according to the accepted Windows directory naming conventions, and subsequently use that alias to access the corresponding resource on the server.

Example 1
The directory alias declared as **news=d:\public\newspapers\news** will allow the clients to receive the file **sports.html** located in the directory **\public\newspapers\news** on the server disk **d:** by issuing the URL **http://web/news/sports.html** from their browsers.

Example 2
The directory alias declaration **news=public\newspapers\news** will allow the clients to receive the file **sports.html** from the sub-directory **\public\newspapers\news** of the server's home directory by issuing the URL **http://web/news/sports.html** from their browsers.

Resource Alias

The resource alias mechanism enables you to substitute a certain name (alias) for any server resource named according to the accepted Windows file naming conventions, and subsequently use this alias to access that resource on the server (as well other resources in the same directory).
While functionally similar to a directory alias, a resource alias also allows you to re-define the default name of a resource (specified in the *HomePage* parameter).

Example 1
The alias declared as **news=d:\public\newspapers\news\sports.html** will allow the clients to receive the file **sports.html** located in the directory **\public\newspapers\news** on the server disk **d:** by issuing the URL **http://web/news** from their browsers. Other files located in that directory will then be also accessible via the same alias, for example: **http://web/news/records.htm**.

Example 2
The alias declaration **news=public\newspapers\news\sports.html** will allow the clients to receive the file **sports.html** from the sub-directory **\public\newspapers\news** of the server's home directory by issuing the URL **http://web/news** from their browsers. Other files located in that sub-directory will then be also accessible via the same alias, for example: **http://web/news/records.htm**.

Name Alias

The name alias mechanism enables you to substitute a certain name (alias) for any file named according to the accepted Windows file naming conventions, and subsequently use this alias to access that file on the server.
In its effect, a name alias is equivalent to a straightforward substitution of a different name for the name of the file requested in the URL.
Every name alias should open with an asterisk (*) to distinguish it from aliases of other types.

Example 1
The alias declaration ***sports.html=sports.exe** allows you to re-define the name **sports.html** in the directory or the alias specified in the URL and equate it to the name **sports.exe.** With this arrangement, the server will run the application program **sports.exe** when it receives a URL that includes the string **public/sports.html**.

Example 2
The alias declaration ***sports.html=web_app\sports.exe** allows you to re-define the name **sports.html** and equate it to the file name **sports.exe** in the sub-directory **web_app** of the directory defined by the *HomeDirectory* parameter or by another alias specified in the URL. With such an arrangement, the server will run the application **d:\Baikonur\web_app\sports.exe** whenever the file **sports.html** is requested (provided that the server's home directory has been defined as **d:\Baikonur**), and will run a program from the subdirectory **web_app** of the directory **public** when **public/sports.html** is specified in the URL.

Example 3
By declaring the alias ***sports.html=c:\web_applications\sports.exe**, you can re-define the name **sports.html** and equate it to the file name **c:\web_applications\sports.exe**. Then, an attempt to request the file **sports.html** will make the server run the application **c:\web_applications\sports.exe**, but will lead to a name conflict if a request like **public/sports.html** is issued.

Format Of Aliases File

The format of the aliases file is similar to that of standard Windows initialization files, and consists of several sections that define aliases for each individual section (virtual server) of the BAIKONUR server.
The general format of the aliases file is as follows:

```
[Alias.Section [.section_name]]
alias_name = link
```

where
section_name
specifies the name of the section in the initialization file; this name must be equivalent to the section name declared in the *AliasSections* parameter of the [Baikonur.Alias] section of BAIKONUR's initialization file (BAIKONUR.INI);

alias_name
specifies the name of the alias; this name may include any characters allowable by URL naming conventions; all blank characters should be replaced with the percent character ("%"), and the percent character proper should be replaced with the double percent character ("%%"); the opening asterisk ("*") in the alias name serves to indicate that this is a file name alias, and is omitted from the name when the server parses the aliases file; during this process, backslash characters ("\") are converted to slash characters ("/"), and a single slash ("/") is interpreted as the alias for the server's home directory (see description of the *AliasHome* parameter); any opening and closing slash ("/") or backslash ("\") characters are ignored and excluded from the alias name;
link
specifies the path to the directory, file, or re-defined name; the file and directory names in the link should have the same format as Windows file names; the names of files located on other workstations of the network are not allowed to be used unless network-mapped disks are available;
When parsing the aliases file, the server checks whether or not any aliases have been defined for the server. If no matching directory or file is found, the alias name is ignored, and a corresponding error message is output to the Log file. The availability of name aliases on the server is not checked unless a direct reference to a file is made.
If the alias link opens with a link to a disk root directory (the slash character "/"), the default system disk (defined in the *HomeDirectory* parameter) is assumed. If the opening slash is missing, then the disk and directory defined in the *HomeDirectory* parameter are assumed. If the names of the server's home directory and its home page have been changed in the aliases file, the interpretation of the alias links relative to the *HomeDirectory* settings changes immediately after the server's home directory and its home page are re-defined.
Alias Definition Examples

```
;Default section aliases
[Alias.Section]
pub = c:\public
appl = d:\web\applications

;FTP aliases
;FTP section should be defined in BAIKONUR.INI
[Alias.Section.FTP]
/ = c:\ftp\public
info = c:\ftp\info
download = d:\pub\download
```

BAIKONUR Clients File

From the point of view of BAIKONUR system, a client (user) is an object described by a set of mandatory and auxiliary attributes. Descriptions of the system clients are stored in a special fixed-format file, which is read and parsed when necessary. By editing individual records in this file, the system administrator can control the server’s response to the requests of both individual clients and groups of clients (through the use of templates). For instance, the administrator can specify a “mailbox” or a logon directory for each individual client.

Clients File Format

The attributes of a client are recorded on separate lines. A line may contain several similar-purpose attributes, but they should be comma-separated.
A comments line may begin with any non-alphabetic character.
The user ID should be unique.

```
“[“user_id”]’
#mandatory parameters
   “name”     “=”  username
   “password” “=”  password
   “account”  “=”  Guests “,” Power Users “,” “@”
194.87.*.*
   ,” “@” 194.88.10-20.*

#optional parameters
   “token”    “=”  Guest:

#auxiliary parameters
   “property” “=”  string
```

A client normally supplies his/her name and password when requested by the browser. Since it is quite possible that several clients might use identical username/password combinations, the “*account*” field is employed to verify the clients’ access rights to the file system resources and to identify clients by their IP addresses in situations when their names coincide. This line contains a list of identifiers for the operating system’s user groups which a client attempting to access a file will be identified as belonging to. BAIKONUR temporarily “adds” such a client to these groups.
The administrator can utilize standard tools (“WinFile” or “Cacls” in Windows NT, or “chmod” in UNIX) to control and manage user access rights. Before sending the contents of a file to the client’s display, BAIKONUR will check the ACL (Access Control List) corresponding to that file in order to verify whether identifiers of the groups specified in the “*Account*” field are listed there. If successful, BAIKONUR will generate a common access mask and display the file’s contents, or return an error message otherwise. One or several masks for the clients’ IP address can be added to the list. Every mask should begin with the commercial “at” character (“@”). The addresses thus added to the ACL will be utilized for client identification purposes only at the username/password verification phase.
If, for instance, the username and password are found to be valid, further search for the client record will be performed based on his/her IP address. This makes it possible to distinguish between “local” and “external” users and specify different server settings for them.
In situations when the clients file may or should include several records on unnamed users (clients that have no name and password), such records will be applied to

non-authorized users in accordance with the address masks. Such unnamed users can be used by the registration program as a template prepared by the system administrator. Suppose, for example, that the administrator has prepared several non-authorized user templates for various groups of IP addresses. Now, a client making a first attempt to access the server's resources without specifying his/her name and password will be checked against one of the non-authorized user templates according to his/her IP address. If such a client wishes to specify his/her name, he/she will have two options: either to supply the attributes of a client known to the system, or else go through the registration procedure. In the latter case, the registration procedure will modify the currently applied non-authorized user template (by adding the new name and password, generating a corresponding unique identifier and, possibly, verifying the IP mask), and will create a new record describing the newly registered client.

Bulky as the above procedure may seem, it does ensure that the resulting record will function satisfactorily until the administrator reviews the list of the newly registered clients and makes appropriate corrections (if necessary).

The "*token*" field is employed for the purposes of authorization of user-initiated processes. In the above example, the processes will be initiated by the user named "*Guest*", which should be "known" to the system. If the "*token*" field is omitted, the user will be able to execute a task with the rights of a service, provided that he/she has the right to invoke the task's file for execution (these rights are determined by the settings of the "*account*" field).

BAIKONUR Server Operating Modes

Depending on its settings and types of the operating and file systems, BAIKONUR server can function in one of four operating modes. These modes are as follows:

1. Fully-fledged operating mode.
2. Functional mode.
3. Debug mode.
4. Personal operating mode.

The *fully-fledged operating mode* is possible only when BAIKONUR server is functioning under control of operating system Windows NT 3.51 or above on disks with a file system that supports verification of file access rights at the system user groups level (such as the NTFS).

With BAIKONUR running in this mode, you can establish and control the rights of clients to read, edit and delete the server's resources down to an individual file, based on specifications of the access rights of the system group which a given client belongs to.

The *functional operating mode* can be implemented when BAIKONUR server is functioning under control of operating system Windows NT 3.51 or above on disks with a file system that supports verification of file access rights at the system level (such as FAT).

In this mode of operation, verification of access rights to the server resources at the operating system level is not supported. However, you can still exercise administration of the clients' access rights at the logical level by making use of Internet address masks, properties of specific clients, and settings of the server's alias system and its resource utilization system (in much the same way as in the fully-fledged operating mode).

When running in the *debug mode*, BAIKONUR does not allow you to control file access rights at system level or use Internet address masks to control clients' access rights. Besides, in this mode the rights of the tasks executing on the server are limited to those of the server proper (thus, if the server functions as a system service, its tasks will likewise have the rights of a service). BAIKONUR changes to the debug mode in the following situations:

1. the server was started with the "-d" switch in the command line;
2. the server was started under control of Windows 95 operating system;
3. the file describing the server's authorized clients (Clients.fil) was not declared, was not found, or was incorrectly specified.

In the *personal operating mode*, there is no pre-defined way to access the server from the outside world via Internet. However, applications that do support the extended data exchange protocol can still execute calls both to another local application on the server machine and to a remote application. Besides, such applications can initiate the server's static protocols to permit TCP/IP-access to the server.

Files Needed To Start BAIKONUR Server

The following files are required to start BAIKONUR server:

1. Server's kernel file (Baikonur.exe).
2. Server's administration tools library (BaikADM.dll), which should be located in the same directory as the Baikonur.exe file.
3. Server's initialization file (Baikonur.ini), which should be located in the home directory of Windows operating system.
4. File describing the server's registered clients (Clients.fil).
5. File describing the system of aliases employed by the server (Alias.fil).
6. File(s) describing the protocol(s) supported by the server (one or more protocol DLLs).
7. SSL certificates file (SERVER.PEM), if the server employs a privacy enhanced mail protocol.

BAIKONUR will terminate the startup sequence if it fails to locate the BaikADM.dll file in the server's startup directory or finds that its version is other than that of the server. BAIKONUR will also terminate the startup sequence if it is unable to find the server's initialization file.

BAIKONUR Server Initialization Sequence

BAIKONUR initialization sequence begins with the server (a) allocating a memory area in which to store its variables and pointers to the functions exported from the server kernel, and (b) parsing the switches included in the startup command line, identifying the operating system type, and determining the startup directory.

Once it determines the operating system type, the server switches to the debug mode (if it runs in Windows 95 environment, the server does so regardless of whether or not the "-d" switch was specified at startup). The startup directory becomes the server's default directory (i.e., the directory where the server will look for the aliases file, the clients file and any additional DLLs until its system directory is specified).

After its variables are initialized to their default values and the administration library

(BaikADM.dll) is located, the server invokes the administration function ADM() from that library, and passes to it an event flag signifying commencement of the server initialization, as well as a pointer to the server's data structure entry point.

The function ADM() compares the server's version with that of the BaikADM.dll library.

If the versions are found to coincide, the function ADM() exports the log file write function from the BaikADM.dll library, and attempts to locate and read the initialization file Baikonur.ini.

If the versions are found to differ or the file Baikonur.ini could not be found, the function ADM() terminates further execution, generates a corresponding error message and returns a zero value to the calling function, with the result that server program gets closed.

The server initialization file (Baikonur.ini) is searched for in Windows home directory (i.e., the directory containing the kernel of Windows NT or Windows 95 operating system).

The file Baikonur.ini is essentially a text file. Its pattern follows the standard format accepted for Windows initialization files. That is, the file is divided into individual sections. The name of each section starts from the beginning of a new line and is enclosed in square brackets. Section parameters are separated from their corresponding values by the equals sign ("="). The semicolon character (";") denotes comments (i.e., anything contained between the comments character and the end of the current line is treated as comments).

After the initialization file is opened, the function ADM() attempts to read and parse its individual sections.

The first section in the Baikonur.ini file should be the [Baikonur] section, where system settings for all subsequent sections of the file should be specified. In addition to that, the [Baikonur] section is employed to specify such information as:

- types of messages to be output to the server's Log file;
- type and name of the server's Log file;
- server's system directory;
- server's home directory;
- name of the authorized clients file;
- name of the SSL certificates file.

Once the server's initialization file has been read, the initializing function attempts to locate and read the authorized clients file (Clients.fil), if the server runs in the debug mode.

The file Clients.fil is searched for either in the server's startup directory (this is also called the server's system directory, and can be re-assigned in the server initialization file), or using the access path to the corresponding files explicitly specified in the Baikonur.ini file.

If the file Clients.fil is not found or none of the server clients is correctly declared there, BAIKONUR will switch to the debug mode (in which any clients are allowed to log on to and register with the server, and no verification of their file access rights is performed).

After all accessible initializing information has been read, the function ADM()

terminates with a positive result, and control is returned to the server. Using the data it receives, the server then attempts to initialize the system of supported protocols in the following sequence:

- creates a window in which to receive messages related to the list of tasks being serviced (task orders list);
- initializes the system's TCP/IP sockets support library (winsock.dll);
- runs the declared number of task-executing threads (also called task handlers, or agents);
- runs the declared number of I/O servicing threads (also called I/O handlers, or managers);
- opens the protocol libraries and searches them for protocol parsers;
- generates orders to process requests arriving to the corresponding ports and adds them to the tasks orders list.

If no protocols are declared in the Baikonur.ini file or no protocol library files are found, or if all the declared protocol ports are already used in the system by other programs, the server switches to the so called personal operating mode (in which it will be accessible only to those local applications which support the extended data exchange protocol).

After initialization is completed and the task servicing system is built, the server again invokes the function ADM() from the Baikonur.dll library using a command line parameter that signifies completion of the initialization process. When invoked with such a parameter, the function ADM() determines the task and function of the system timer, opens the file describing the system of aliases (Alias.fil), and attempts to interpret it relative to the directory declared as the server's home directory.

If the file Alias.fil cannot be found or opened, there will exist only a root alias (coinciding either with the server's home directory as it is declared in the [Baikonur] section of the initialization file or, if no home directory is declared, with the system directory) for all the server sections declared in the Baikonur.ini file.

Once it finishes reading the aliases file, the function ADM() terminates and returns a corresponding code, which the server then uses to either switch to the active state or shut down.

How To Organize Access To Server From Network

All you need do to organize access to the server from a network is to connect the computer with Windows NT operating system (or Windows 95, although this is not recommended) installed on it to Internet or your local network, install the BAIKONUR software, and make the appropriate settings. This is practically all it takes to start using BAIKONUR as a web server. If you also want to be able to develop BAIKONUR applications, you will need Borland Delphi or Borland C++ Builder.

Contact Your ICP To Gain Access To Internet

To get your BAIKONUR Web server connected to Internet, contact your Internet service provider (ISP). The provider will give you the IP address of your server, the sub-network mask, and other essential attributes.

If you intend to employ BIAKONUR merely as an Intranet server in your local network, there is no need to resort to the services of an Internet service provider, for you can make all the necessary settings yourself.

Before You Install BAIKONUR

If you already have a suitable operating system (like Windows NT Workstation, Windows NT Server or Windows 95) installed on your computer, you can install BAIKONUR from the installation diskettes. Or you may prefer to copy the contents of these diskettes to your hard disk and perform installation from there. If your computer is already connected to Internet, information from your Web server will be available to other Internet users immediately after you complete the installation procedure and set up BAIKONUR's directories and aliases as necessary. Normally, most of the default settings are acceptable for your information to be readily accessible to Internet/Intranet users without any further modifications.

This part of the guide describes general requirements to BAIKONUR installation and explains how you can best configure your operating system in preparation to this installation.

Requirements To Installation

To install BAIKONUR, you will need to have the following:

- Computer with a minimal configuration (sufficient to ensure proper functioning of the operating system)
- Windows NT Workstation, version 3.51 or (optimally) 4.0, or Windows 95 (if you work at home), of Windows NT Server 3.51 or 4.0.
- Transmission Control Protocol/Internet Protocol (TCP/IP). This protocol is a standard component of all the operating systems mentioned above. To set up and configure the TCP/IP protocol, open the Network folder in the Control Panel.
- A floppy-disk drive (if you install BAIKONUR from diskettes) or a copy of BAIKONUR's installation diskettes on your hard disk.
- Enough free space on your hard disk to accommodate all your BAIKONUR Web server's files and information and support its log files. We strongly recommend that you use Windows NT File System (NTFS) to allocate the files used by BAIKONUR.

To have your information published in Intranet, you will need to have the following:

- A network adapter card and a connection to your local network.
- A Windows Internet Name Service (WINS) or Domain Name System (DNS) server installed on the Intranet computers. Although this step is optional, it does allow the network clients to employ a friendly and easy-to-understand name system instead of having to fiddle with numeric IP addresses.

To have your Web site published in Internet, you must do the following:

- Establish a connection with Internet and post the IP address supplied by your Internet service provider (ISP).

- Register your IP address with the DNS. This step is optional, but it will allow your clients to use a friendly and easy-to-understand name system rather than the rather awkward numeric IP addressing technique. For example, *demo.ru* is the registered domain name allocated to our *Epsylon Technologies* company in Internet. Within this domain, we chose *www.demo.ru* to be the name of our World Wide Web (WWW) server.

Most of ISPs can register a domain name for you.

Configuring Windows NT

You must configure the network settings of your Windows NT system so that your Web server could work in the network. We recommend that you use more serious security settings than those set by default. This is important if you want to preclude attempts of an unauthorized access to your server.

Most of the settings we shall speak about below can be viewed and modified with the help of the *Network* application program in the Control Panel.

Configuring TCP/IP Protocol

Install the TCP/IP protocol for your operating system and the necessary communication utilities.

Thus, if you use a Dial-Up IP, you will need to install the utility for dialing to your Internet service provider's host machine. If you wish to make your computer accessible via Internet, your service provider must allocate a dedicated IP address, a sub-network mask and the gateway computer's IP address for your server. By default, it is assumed that the service provider's computer via which your server computer receives all Internet traffic shall be the gateway computer.

NOTE. If any Internet services had already been installed on your computer, either delete them or thoroughly study your BAIKONUR server's administration capabilities to avoid possible conflicts. Although BAIKONUR server is compatible with other vendors' Internet software (such as Microsoft Information Server or Netscape FastTrack Server) and can run concurrently with them on one and the same machine without conflicts, this option requires careful configuration of the entire system on your part.

Choose a domain name (also called *host name*) for your Web site. Of course, your Web server can also be accessed by its IP address (for instance, like http://192.177.55.12/default.htm), but it is much more convenient to do so using a registered domain name (for example, like http://www.demo.ru/home.htm). Ask your Internet service provider to register your domain name.

Configure your system so that your domain name would correspond to the IP address of your computer. Once you do so, Internet users will be able to address your Web server simply by entering your server's domain name in the *Location* field of their browsers.

In Intranet, you may use both the DNS and WINS names for addressing your server. However, your network should then include computers with the appropriate server (DNS or WINS) installed on them, and the client computers should "know" the IP address of that server to establish a connection. As an alternative to DNS and WINS servers, you can use the file HOSTS or LMHOSTS, respectively.

To appropriately configure the TCP/IP protocols and make the name of your computer correspond to its network address, use the *Network* application in the Control Panel.
In this part of the guide we discuss the basic requirements that virtually all Web servers should met while working in a TCP/IP network.

Routing Programs and Security Tools

TCP/IP is a routable protocol, which means that every part of the information exchange system has a specific node address in the network to which data is to be routed. There exist specialized routing programs designed to connect two networks and manage the exchange of packets between them. These routing programs verify the destination address for each packet in one network and, should the addressee happen to be in another network, route the packet to it.

The routing programs can be configured so that only certain packets would be allowed to pass through the bridge between the networks. This process is called *packet screening*. Packet screening can be used, for example, to deny outsiders access to the internal computers and resources.

If you use a TCP/IP network, you most probably already have network routing programs installed. In some case, the Internet service provider installs a routing program between Internet and your workstation, and you can use that to screen the incoming and outgoing packets. Please read the relevant documentation to learn more on how to configure routing programs and similar network security tools.