**File Transfer Protocols**

# What is File Transfer?

File transfer is the process of **sending or receiving files between two systems** over a network, commonly used for B2B communication, automation, and data exchange.

## Common Protocols

| **Protocol** | **Secure** | **Port** | **Used For** |
| --- | --- | --- | --- |
| **FTP** | ❌ No | 21 | Legacy systems |
| **FTPS** | ✅ Yes | 990 / 21 | FTP + SSL |
| **SFTP** | ✅ Yes | 22 | Secure, modern standard |
| **HTTP(S)** | ✅ Yes | 80/443 | File transfer via API |
| **SMB/NFS** | Internal | OS Dependent | Shared folders in LAN |

## File Transfer Concepts

* **Client/Server model**: One sends, the other receives
* **Authentication**: Username/password, private key, or certificate
* **Transfer modes**: Binary (for media/files) vs ASCII (for text)
* **Polling vs Scheduled**: Time-based vs event-driven automation

## Common Operations

| **Operation** | **Description** |
| --- | --- |
| Connect | Login/authenticate to remote server |
| Put | Upload file |
| Get | Download file |
| List | List remote directory contents |
| Delete | Remove file |

Automation Types

* **Scheduler-based**: Run every hour/day/week (time-driven)
* **File Polling**: Automatically process files dropped in folder (event-driven)
* **Streaming**: Process large files without loading into memory
* **PGP Encryption**: Secure the content of sensitive files

# How File Transfer Is Done in webMethods

## Services Used

| **Protocol** | **Service Package** | **Example Service** |
| --- | --- | --- |
| FTP | pub.client.ftp | ftp:login, put, get |
| SFTP | pub.client.sftp | sftp:login, put, get |
| Polling | File Polling Port | Configured in IS Admin |
| Scheduling | IS Scheduler | Settings > Scheduler |

## Flow Elements to Use

* **Connect**: sftp:login / ftp:login
* **Upload**: sftp:put / ftp:put
* **Download**: sftp:get / ftp:get
* **Logout**: sftp:logout / ftp:logout
* **File handling**: pub.file:move, pub.file:getFile
* **Logging**: Custom logging or pub.flow:getLastError
* **PGP Encryption**: External Java service or OS shell command

## Automation in webMethods

* **Scheduler**: Trigger Flow service at fixed time
* **File Polling**: Automatically run service when file arrives
* **Archiving**: Use pub.file:rename or move service
* **Notifications**: Use pub.client:smtp for email alerts

# Real-Time Telecom Example: Usage File Sent to Billing System via SFTP (Hourly)

## 🔹 Scenario:

The telecom system collects **data usage records** every hour and sends them to the **billing system** using **SFTP**.  
The billing system uses this data to calculate and update customer bills.

General Workflow (Conceptual)

1. Generate .csv file with usage records
2. Connect to SFTP using key or password
3. Upload file to billing system folder
4. Archive the file locally
5. Log status (success or fail)
6. Repeat every hour using a scheduled job

## File Example:

usage\_20250619\_1000.csv

MSISDN,UsageType,UnitsUsed,Timestamp

9876543210, DATA,156,2025-06-19 10:00:00

9911223344, VOICE,120,2025-06-19 10:01:00

## How It’s Done in webMethods

## Step 1: Create Scheduled Job

* IS Admin → Settings → Scheduler
* Schedule service: sendUsageFileToBilling
* Run every hour (e.g., HH:01)

## Step 2: Flow Service: sendUsageFileToBilling

1. **Get Latest File**
   * Use pub.file:getFileList to pick correct file
   * Build file name dynamically based on system date/time
2. **(Optional) Encrypt or Compress**
   * Use Java service or execOSCommand for GPG or gzip
3. **Connect to SFTP**
   * pub.client.sftp:login
     + Host: sftp.billing-system.com
     + Auth: username/password or key
4. **Upload File**
   * pub.client.sftp:put
     + Local file: /data/usage/usage\_20250619\_1000.csv
     + Remote folder: /incoming/usage/
5. **Logout**
   * pub.client.sftp:logout
6. **Archive File**
   * pub.file:move
     + From: /data/usage/
     + To: /archive/usage/20250619/
7. **Log Transfer**
   * Log to DB or flat file (status, timestamp, file name)
   * Use pub.flow:getLastError for error handling

## Optional Enhancements

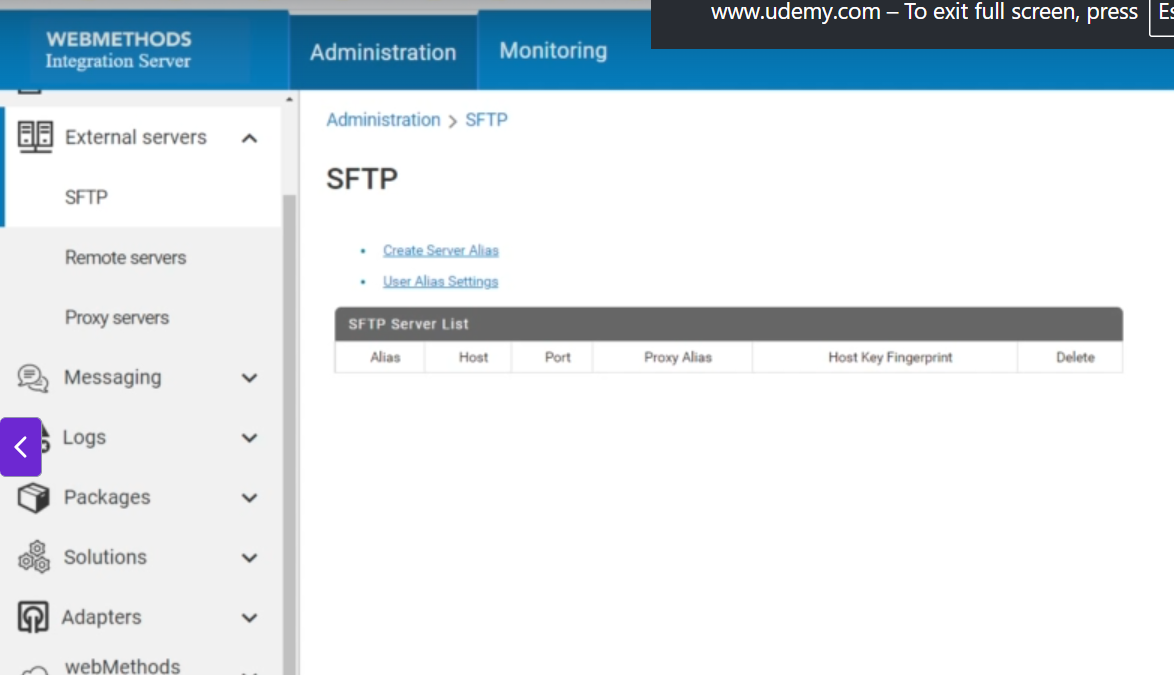
* Retry logic if upload fails
* Email alert for failure using pub.client:smtp
* Create audit report for all hourly transfers

## Interview-Safe Answer

In my telecom project, I implemented an integration to send hourly usage files to the billing system.  
I scheduled a Flow service to pick up usage CSV files, connect to SFTP securely, upload them, and archive the files.  
I also added error handling and logging to monitor the success of each transfer.

Of course, Sankar! Here's the **fully rewritten version** — clean, simple, and structured like a note you'd keep for interviews, learning, or explaining to others.

# How to Connect to an SFTP Server in webMethods

To connect to an SFTP server from webMethods Integration Server, we need to configure **two things**:  


## 1️⃣ SFTP Server Alias

**Purpose**: Defines **where to connect** (the server's details).

**It includes:**

* SFTP server hostname or IP address
* Port number (default is 22)
* Host key (for verifying the server identity)

💡 Think of this as setting the **address of the server**.

## 2️⃣ SFTP User Alias

**Purpose**: Defines **how to connect** (the user credentials and connection behavior).

**It includes:**

* SFTP username
* Authentication method (password or public/private key)
* Password or private key location
* Server alias reference (linking to the server alias above)
* Optional settings: retry count, timeouts, compression, etc.

💡 Think of this as setting the **login credentials and client config**.

## ❓ Why Two Aliases? Why Not Just One?

| **Reason** | **Explanation** |
| --- | --- |
| ✅ **Reusability** | One server alias can be reused by multiple user aliases (if you have different users for the same server). |
| ✅ **Separation of Concerns** | Keeps server details (host, port, keys) separate from user authentication. Easier to maintain. |
| ✅ **Flexibility** | If the server’s IP or host key changes, you only update the server alias. All linked user aliases still work. |
| ✅ **Security** | Sensitive login details are kept in the user alias, not mixed with general server info. |
| ✅ **Clarity in Configuration** | Makes things organized and easy to manage inside Integration Server Administrator. |

## 🧪 Real-World Example:

You have one SFTP server:  
192.168.1.100 on port 22  
That’s your **SFTP Server Alias** — let’s name it: MySFTPServer

Now you want to connect with two users:

* user1 using **password**
* user2 using **public key**

So you create:

* **User Alias 1** → User\_Alias\_One with user1 + password
* **User Alias 2** → User\_Alias\_Two with user2 + public/private key

Both user aliases **point to the same server alias**, MySFTPServer.

Let me know if you want this in diagram format or a one-page PDF-style note for quick revision.

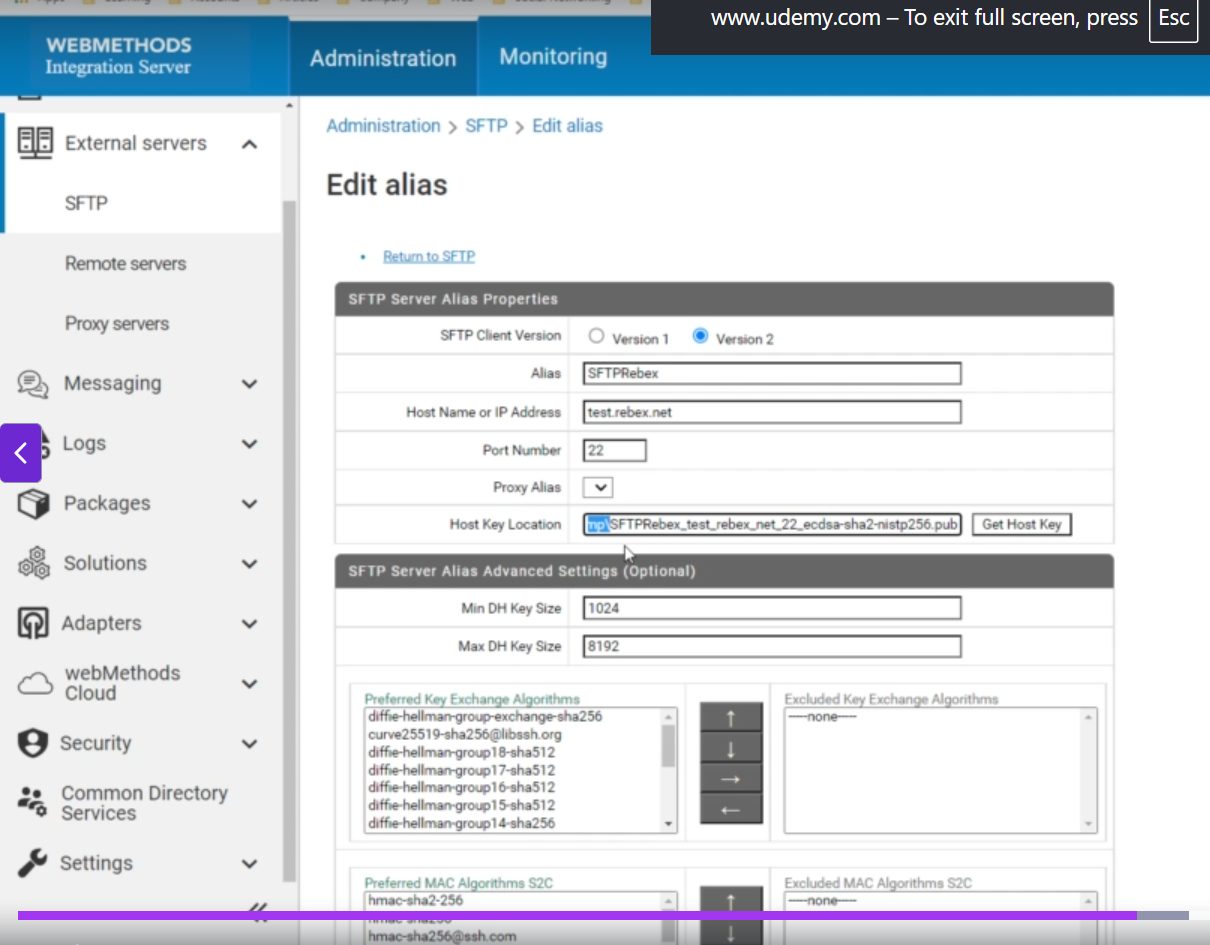
## Step-by-Step: How to Create an SFTP Server Alias in webMethods

**🔷 What Is an SFTP Server Alias?**

An **SFTP server alias** in webMethods is a **saved configuration profile** that tells Integration Server:

* Which SFTP host to connect to
* On which port
* How to validate the server using its **host key**
* Whether to use any proxy

🔐 It’s **mandatory** to create this before you can create a user alias or perform any file transfer.



**✅ Step-by-Step Guide**

**🔹 Step 1: Open IS Admin**

* URL: http://<host>:5555
* Login with Admin credentials

**🔹 Step 2: Go to External Servers > SFTP**

* In the left panel, find:
* Security → External Servers → SFTP
* Click on: **Create Server Alias**

**🔹 Step 3: Fill in Server Alias Configuration**

| **Field** | **What to Fill** |
| --- | --- |
| **SFTP Client Version** | Choose **Version 2** (Recommended; more options) |
| **Alias** | Give a unique name (e.g., BillingSFTPServer) |
| **Host Name/IP** | The hostname of the SFTP server (e.g., sftp.example.com) |
| **Port Number** | Usually 22 |
| **Proxy Alias** *(optional)* | Leave blank if you're not using a proxy (recommended for internal servers) |
| **Host Key Location** | This is where you provide the **public key file path** of the server |

**🔹 Step 4: Get the Host Key Automatically (Optional but Easy)**

If you don't already have the SFTP server’s public key file:

1. Click the **"Get Host Key"** button
2. IS will connect to the server using the host & port
3. It retrieves the **public key** and stores it temporarily
4. It auto-fills the **Host Key Location** field

✅ This is a safe and quick way to get the key — and confirms the server is reachable.

**🔹 Step 5: Save the Configuration**

* Click **Save**
* You’ll now see the server alias listed

✅ 🎉 Your SFTP Server Alias is ready!

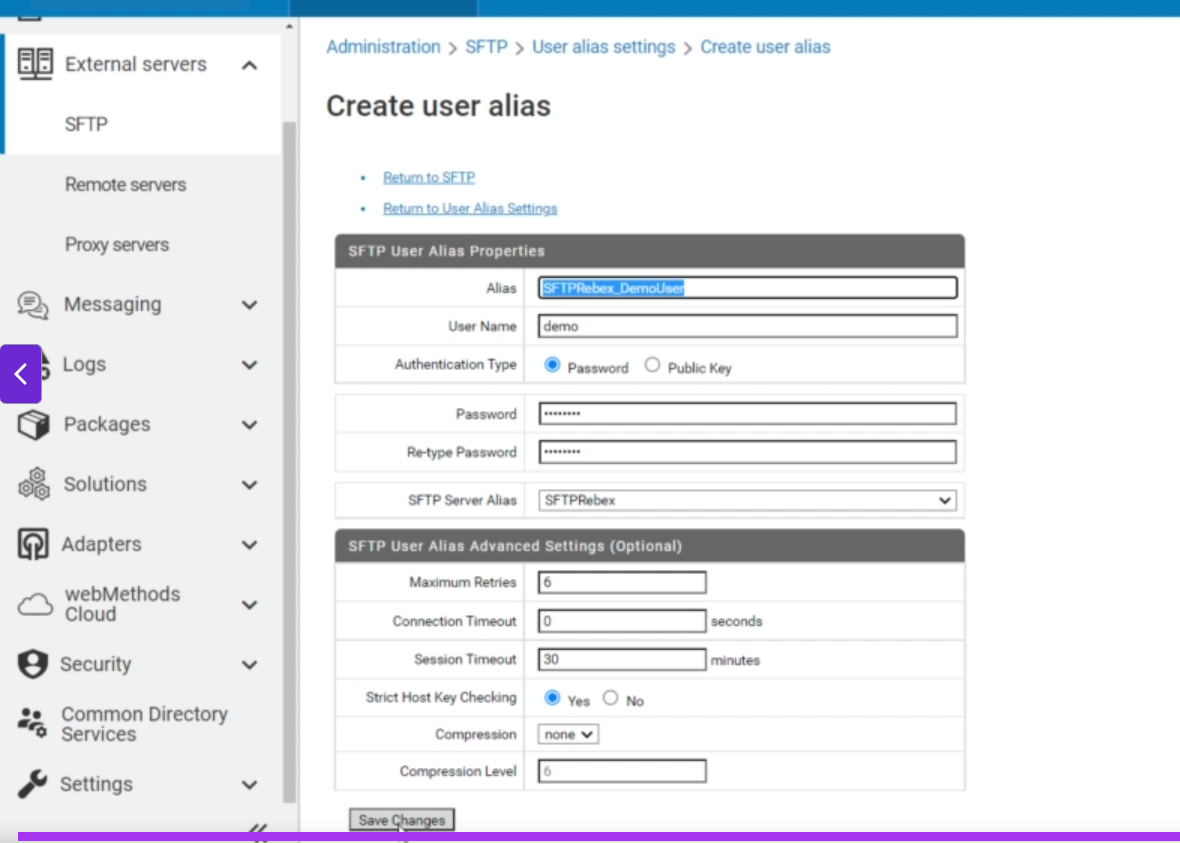
**🧠 Example Configuration for Demo/Test**

| **Parameter** | **Value** |
| --- | --- |
| Client Version | Version 2 |
| Alias | RebexSFTPServer |
| Host | test.rebex.net |
| Port | 22 |
| Host Key Location | Use **Get Host Key** button |

## Creating an SFTP User Alias in webMethods

An **SFTP User Alias** stores the login credentials and connection settings Integration Server (IS) uses to connect to an SFTP server as a client.

🔹 You **must create an SFTP Server Alias first** before creating a User Alias.



**🔧 Steps to Create SFTP User Alias:**

1. **Open Integration Server Administrator**
2. Navigate to: External Servers > SFTP
3. Click **User Alias Settings**
4. Click **Create User Alias**

**🔑 Key Fields and Parameters:**

| **Parameter** | **Description** |
| --- | --- |
| **Alias** | Unique name for the user alias (only \_ and . allowed; max 255 chars; can't start with http://) |
| **User Name** | SFTP account username |
| **Authentication Type** | Choose either:🔸 Password🔸 Public Key |
| **Password / Re-type Password** | (If using password authentication) |
| **Private Key Location** | (If using public key auth) Path to the private key |
| **Passphrase / Re-type Passphrase** | (If private key is encrypted) |
| **SFTP Server Alias** | The related Server Alias created earlier |
| **Maximum Retries** | Retry attempts (1–6, default: 6) |
| **Connection Timeout (seconds)** | Timeout for connection attempt (default: 0 – no timeout) |
| **Session Timeout (minutes)** | Idle session timeout (default: 10 mins) |
| **Strict Host Key Checking** | Choose: ✅ Yes – verifies SFTP host key❌ No – skips verification *(Recommended: Yes)* |
| **Compression** | Options: 🔸 zlib – compress data 🔸 None – no compression |
| **Compression Level** | (Only if zlib used) Range: 1 (fast) to 6 (max compression). Default: 6 |

**📁 Config File Location:**

The alias gets saved at:  
IntegrationServer\_directory/instances/<instance\_name>/config/sftp/sftpUserAliases.cnf

Great, Sankar. Based on your requirement, here is the **detailed documentation of SFTP services in webMethods**, organized from **most important to least important**, written in a clean and professional format:

# webMethods SFTP Inbuilt Services

**Package:** WmPublic → pub.client.sftp and pub.client.sftp.admin  
**Purpose:** Enables Integration Server to securely connect to external SFTP servers to perform file transfers, directory operations, and session management.

**🔶 Tier 1 – Core & Most Used Services**

## pub.client.sftp:login

**Description:**  
Establishes a secure connection to an SFTP server. Returns a sessionKey, which is required for all subsequent SFTP operations.

**Key Inputs:**

* userAlias: (Recommended) Alias to preconfigured SFTP settings.
* reuseSession: (Optional) true to reuse open session; default is false.
* sftpConfigurationParameters: (Optional) Manual SFTP connection settings like host, port, username, authenticationType, etc.

**Key Outputs:**

* sessionKey: Unique identifier for this session.
* returnCode, returnMsg: Status of login attempt.

**Usage Notes:**  
Always call this before using other SFTP services.  
Auto-logout can be controlled using watt.client.sftp.session.logoutOnServiceCompletion.

## pub.client.sftp:put

**Description:**  
Uploads files from Integration Server to the SFTP server.

**Key Inputs:**

* sessionKey
* contentStream: File stream (optional)
* localFile: Local file name (optional)
* remoteFile: Destination path on SFTP
* mode: overwrite (default), append, or resume

**Key Outputs:**

* returnCode, returnMsg

**Usage Notes:**

* If using contentStream, remoteFile is required.
* Use for outbound file integrations.

## pub.client.sftp:get

**Description:**  
Downloads a file from the SFTP server.

**Key Inputs:**

* sessionKey
* remoteFile: Path on SFTP
* localFile: (Optional) Local destination
* mode: overwrite, append, or resume

**Key Outputs:**

* contentStream: File stream (if localFile not given)
* returnCode, returnMsg

**Usage Notes:**

* Commonly used in inbound file integrations.
* Use localFile to save to disk, or handle the stream for in-memory processing.

## pub.client.sftp:logout

**Description:**  
Ends the SFTP session associated with the given sessionKey.

**Key Inputs:**

* sessionKey

**Key Outputs:**

* returnCode, returnMsg

**Usage Notes:**

* Always logout manually if auto-logout is disabled.

## pub.client.sftp:ls

**Description:**  
Lists files in a directory on the SFTP server.

**Key Inputs:**

* sessionKey
* path: Optional. Supports \* wildcard in version 2

**Key Outputs:**

* dirList[]: Contains fileName, fileSize, permissions, modifiedTime, uid, gid, etc.
* returnCode, returnMsg

**Usage Notes:**  
Useful for checking file existence before download or deletion.

**🔷 Tier 2 – Operational & Housekeeping Services**

## pub.client.sftp:mkdir

**Description:**  
Creates a new directory on the SFTP server.

**Inputs:**

* sessionKey
* path: Path to be created

**Outputs:**

* returnCode, returnMsg

**Usage Notes:**  
Often used before uploading files into organized folders.

## pub.client.sftp:rm

**Description:**  
Deletes a file from the SFTP server.

**Inputs:**

* sessionKey
* path: File to be deleted

**Outputs:**

* returnCode, returnMsg

**Usage Notes:**  
Good for cleaning up processed files.

## pub.client.sftp:rmdir

**Description:**  
Deletes one or more directories.

**Inputs:**

* sessionKey
* path: Must be empty folder

**Outputs:**

* returnCode, returnMsg

**Usage Notes:**  
Used for post-processing cleanup.

## pub.client.sftp:rename

**Description:**  
Renames or moves a file/folder.

**Inputs:**

* sessionKey
* oldPath, newPath

**Outputs:**

* returnCode, returnMsg

**Usage Notes:**  
Useful for archiving or renaming after processing.

## pub.client.sftp:pwd

**Description:**  
Returns the current working directory on the remote server.

**Inputs:**

* sessionKey

**Outputs:**

* path, returnCode, returnMsg

**Usage Notes:**  
Helpful during troubleshooting or relative path operations.

**🟣 Tier 3 – Advanced/Admin or Rarely Used**

## pub.client.sftp:cd

**Description:**  
Changes the current working directory on the server.

**Inputs:**

* sessionKey, path

**Outputs:**

* returnCode, returnMsg

## pub.client.sftp:chmod

**Description:**  
Changes file permission (e.g., 644, 777).

**Inputs:**

* sessionKey, mode, path

**Outputs:**

* returnCode, returnMsg

## pub.client.sftp:chgrp

**Description:**  
Changes the group ownership of a file.

**Inputs:**

* sessionKey, groupId, path

**Outputs:**

* returnCode, returnMsg

## pub.client.sftp:chown

**Description:**  
Changes user ownership of a file.

**Inputs:**

* sessionKey, uid, path

**Outputs:**

* returnCode, returnMsg

## pub.client.sftp:symlink

**Description:**  
Creates a symbolic link to a file.

**Inputs:**

* sessionKey, oldPath, newPath

**Outputs:**

* returnCode, returnMsg

**🛠️ Admin Utilities (Optional, but Useful)**

## pub.client.sftp.admin:getDefaultAlgorithms

**Purpose:**  
Returns the default list of supported algorithms, MACs, and ciphers used by the SFTP client.

**Use Case:**  
Useful when configuring security settings or troubleshooting handshake issues.

## pub.client.sftp.admin:getHostKey

**Purpose:**  
Retrieves the host key of the remote SFTP server for verification.

**Use Case:**  
Required when using strict host key checking in secure production environments.

## General Notes

* **Parallel Calls:** SFTP services cannot be run in parallel using the same sessionKey.
* **Client Version:** Use sftpClientVersion = version2 for modern algorithm and cipher support.
* **Security Tip:** Always use userAlias for configuration — more secure and manageable.

Great, Sankar! Since you've now provided all the FTP-related service documentation in **webMethods**, I’ll structure your **FTP service notes** just like I did for the SFTP documentation — ordered by **most important to least important**, clearly grouped, and with **clean and practical explanations** for each.

# FTP in webMethods – Built-in Services

## 1. pub.client.ftp:login

**Purpose:** Establishes a connection and logs into the remote FTP server.

* **Must be called before most FTP operations.**
* Returns a sessionkey which is used in subsequent calls.
* Supports **passive/active** transfer modes.
* Supports **secure FTP** via auth → None, SSL, TLS, TLS-P.

**Key Inputs:**

* serverhost, serverport, username, password
* transfertype – passive or active (default is active)
* encoding – character set for session (e.g., UTF-8)
* secure.auth – for secure FTP (TLS/SSL)

**Output:**

* sessionkey, returncode, returnmsg, logmsg

## pub.client.ftp:logout

**Purpose:** Terminates the current FTP session using the sessionkey.

* Must be used **explicitly** unless watt.client.ftp.session.logoutOnServiceCompletion=true.

**Input:** sessionkey

## pub.client.ftp:get

**Purpose:** Downloads a file from the FTP server.

* Supports **streaming** for large files.
* Can write to disk or return file as bytes.

**Important Inputs:**

* sessionkey, remotefile
* localfile (optional)
* largefilethreshold – if exceeded, content returned as contentstream
* transfermode (ascii/binary), encoding

**Outputs:**

* content or contentstream, returncode, islargefile

## pub.client.ftp:put

**Purpose:** Uploads a file to the remote FTP server.

* Supports putunique flag for **STOU** (unique name).

**Inputs:**

* sessionkey, remotefile
* Either localfile or content
* transfermode, putunique

**Outputs:**

* returncode, returnmsg

## pub.client:ftp

**Purpose:** All-in-one operation for login, cd, get/put/ls, and logout.

* Easy for simple operations but **less flexible** than session-based methods.
* Can also work with secure FTP via the secure parameter.

**Recommended for:** quick file transfers without needing session reuse.

## pub.client.ftp:cd

**Purpose:** Changes directory on FTP server.

* Needed before file operations if not in default dir.

**Input:** sessionkey, dirpath

## pub.client.ftp:dir

**Purpose:** Lists file names **with timestamps and file info**.

* Equivalent to FTP dir command.

**Input:** filenamepattern (e.g., \*.txt)

## pub.client.ftp:ls

**Purpose:** Lists file names only.

* Uses FTP NLST command.
* Can be ordered by timestamp.

## pub.client.ftp:cdls

**Purpose:** Changes directory **and** lists files.

* Useful combo to avoid multiple calls.

## pub.client.ftp:delete

**Purpose:** Deletes a single file.

**Input:** remotefile

## pub.client.ftp:mdelete

**Purpose:** Deletes multiple files by pattern (e.g., \*.log)

* Warning: If pattern not provided, it may delete **all files**.

## pub.client.ftp:mget

**Purpose:** Retrieves multiple files by pattern.

* Saves files to a local directory.
* Doesn't support in-memory file return like get.

## pub.client.ftp:mput

**Purpose:** Sends multiple files from a local directory to FTP server.

* Useful for bulk uploads.
* Respects putunique.

## pub.client.ftp:append

**Purpose:** Appends data to a file on the FTP server.

* If the file doesn’t exist, it is created.
* Use content or localfile.

## pub.client.ftp:rename

**Purpose:** Renames a file on the FTP server.

**Inputs:** oldname, newname

pub.client.ftp:quote

**Purpose:** Executes a **custom FTP command** directly.

* Useful for non-standard operations.

## pub.client.ftp:sessioninfo

**Purpose:** Retrieves details of all active FTP sessions.

**Output:** List of active session details like serverhost, username, dataport, etc.

## pub.client.ftp:getCompletedNotification

## pub.client.ftp:putCompletedNotification

**Purpose:**

* Publishable document types used internally by **Integration Server FTP Listener**.
* Triggered after get or put commands **complete**.
* Meant for **local-only publish/subscribe** use.

## 🔚 Summary: Most Used FTP Services for Common Tasks

| **Use Case** | **Service(s) to Use** |
| --- | --- |
| Simple get/put one file | pub.client.ftp:get / pub.client.ftp:put |
| Simple get/put without sessions | pub.client:ftp |
| List files | pub.client.ftp:ls / dir |
| Delete file(s) | delete, mdelete |
| Multi file upload/download | mput, mget |
| Directory change/listing | cd, cdls |
| Session start/close | login, logout |
| Append data to remote file | append |