Appendix A - Readme file

Clustering Source code comments

System Requirements

- Python 3.6 (pythonw (Python GUI) is required to view the plot results)
- pip3 (for installing python dependent modules)

Python Dependencies

- nltk
- pandas
- sklearn
- colorma
- javalang
- python-dateutil

Preparing Environment

- Unzip or clone (`git clone https://github.com/wsimf/classify_source_code_comments.git`) the project to a convenient location
- Open the command prompt / terminal and navigate to the project folder
- Make sure Python3 is installed (`python -V`)
- Install pythonw (if you are using anaconda for python environment management, you can use `conda install python.app` on a macOS environment)
- Install dependencies (`pip install -r requirements.txt`)

Running the application

- Copy Java source files into `/comment_resources` directory
- Run `pythonw comment_analyser.py`
- All the found comments will be written into `comments.txt` file
- The clustered result will be written into `comments_cluster_dbscan.txt` file
- The count and TF-IDF vectorizer results will open in a seperate window

Python GUI (plot results) were tested on a macOS environment. The `pythonw` should be installed depending on the operating system used. Not all operating systems support python GUI. The program would fail with an error if it was unable to draw the GUI component. However, the clustering and the generation of `comments.txt` and `comments_cluster_dbscan.txt` will successfully complete irrespective of this issue.

APPENDIX B - CLUSTER RESULT 01

Some parts of the cluster results file is attached below. For the full result list, please refer to the attached `comments_cluster_dbscan.txt` file.

```
** [CLUSTER] : 0
      [COMMENT] : Verify the LOHS onStarted callback is triggered when WifiManager receives a HOTSPOT_STARTED message from
WifiServiceImpl. /
      [COMMENT] : Verify onStopped is called if WifiServiceImpl sends a HOTSPOT_STOPPED message. /
      [COMMENT] : Verify the LOHS onRegistered observer callback is triggered when WifiManager receives a HOTSPOT_OBSERVER_REGISTERED
message from WifiServiceImpl. /
      [COMMENT] : Verify the LOHS onStarted observer callback is triggered when WifiManager receives a HOTSPOT_STARTED message from
WifiServiceImpl. /
      [COMMENT] : Verify the LOHS onStarted observer callback is triggered not when WifiManager receives a HOTSPOT_STARTED message from
WifiServiceImpl with a null config. /
      [COMMENT] : Verify the LOHS onStopped observer callback is triggered when WifiManager receives a HOTSPOT_STOPPED message from
WifiServiceImpl. /
      [COMMENT] : Verify WifiServiceImpl is not called if there is not a registered LOHS observer callback. /
      [COMMENT] : Verify WifiServiceImpl is called when there is a registered LOHS observer callback. /

** [CLUSTER] : 1
      [COMMENT] : Verify that the call to cancel a LOHS request does call stopLOHS. /
      [COMMENT] : Verify that the callback is not triggered if the LOHS request was already cancelled. /
      [COMMENT] : Verify that calling cancel LOHS request does not crash if an error callback was already handled. /

** [CLUSTER] : 2
      [COMMENT] : Validate the successful connect flow: (1) connect + success (2) publish, (3) disconnect (4) try publishing on old
session (5) connect again /
      [COMMENT] : Validate the publish flow: (0) connect + success, (1) publish, (2) success creates session, (3) pass through everything,
(4) update publish through session, (5) terminate locally, (6) try another command - ignored. /
      [COMMENT] : Validate the subscribe flow: (0) connect + success, (1) subscribe, (2) success creates session, (3) pass through
everything, (4) update subscribe through session, (5) terminate locally, (6) try another command - ignored. /
      [COMMENT] : Validate ranging + success flow: (1) connect, (2) create a (publish) session, (3) start ranging, (4) ranging success
callback, (5) ranging aborted callback ignored (since listener removed). /

** [CLUSTER] : 3
      [COMMENT] : Validate that an empty agent network specifier doesn't match any base network specifier. /
      [COMMENT] : Validate that an agent network specifier constructed with a single entry matches that entry, and only that entry. /
      [COMMENT] : Validate that an agent network specifier constructed with multiple entries matches all those entries - but none other. /
      [COMMENT] : Validate that agent network specifier does not match against a sub-set. /

** [CLUSTER] : 5
      [COMMENT] : Helper function for creating a {@link Policy} for testing.  @return {@link Policy} /
      [COMMENT] : Helper function for creating a {@link PasspointConfiguration} for testing.  @return {@link PasspointConfiguration} /
      [COMMENT] : Helper function for generating certificate credential for testing.  @return {@link Credential} /
      [COMMENT] : Helper function for generating SIM credential for testing.  @return {@link Credential} /
      [COMMENT] : Helper function for generating user credential for testing.  @return {@link Credential} /
      [COMMENT] : Helper function for creating a {@link UpdateParameter} for testing.  @return {@link UpdateParameter} /

** [CLUSTER] : 6
      [COMMENT] : Verify that copy constructor works when pass in a null source.  @throws Exception /
      [COMMENT] : Verify that copy constructor works when pass in a valid source.  @throws Exception /
      [COMMENT] : Verify that copy constructor works when pass in a source with user credential.  @throws Exception /
      [COMMENT] : Verify that copy constructor works when pass in a source with certificate credential.  @throws Exception /
      [COMMENT] : Verify that copy constructor works when pass in a source with SIM credential.  @throws Exception /
      [COMMENT] : Verify that policy created using copy constructor with null source should be the same as the policy created using
default constructor.  @throws Exception /
      [COMMENT] : Verify that policy created using copy constructor with a valid source should be the same as the source.  @throws
Exception /
      [COMMENT] : Verify that a policy created using {@link #createPolicy} is valid, since all fields are filled in with valid values.
@throws Exception /
      [COMMENT] : Verify that UpdateParameter created using copy constructor with null source should be the same as the UpdateParameter
created using default constructor.  @throws Exception /
      [COMMENT] : Verify that UpdateParameter created using copy constructor with a valid source should be the same as the source.
@throws Exception /
      [COMMENT] : Verify that an UpdateParameter created using {@link #createUpdateParameter} is valid, since all fields are filled in
with valid values.  @throws Exception /

** [CLUSTER] : 7
      [COMMENT] : Verify a valid installation file is parsed successfully with the matching contents.  @throws Exception /
      [COMMENT] : Verify that parsing an installation file with invalid MIME type will fail.  @throws Exception /
      [COMMENT] : Verify that parsing an un-encoded installation file will fail.  @throws Exception /
      [COMMENT] : Verify that parsing an installation file that contains a non-base64 part will fail.  @throws Exception /
      [COMMENT] : Verify that parsing an installation file that contains a missing boundary string will fail.  @throws Exception /
      [COMMENT] : Verify that parsing an installation file that contains a MIME part with an invalid content type will fail.  @throws
Exception /
      [COMMENT] : Verify that parsing an installation file that doesn't contain a Passpoint profile will fail.  @throws Exception /

** [CLUSTER] : 8
      [COMMENT] : Helper function for creating a map of home network IDs for testing.  @return Map of home network IDs /
      [COMMENT] : Helper function for creating a HomeSp for testing.  @param homeNetworkIds The map of home network IDs associated with


** [CLUSTER] : 4
      [COMMENT] :  Setting the client certificate to null should clear the existing chain.
      [COMMENT] : Verify parcel read/write for an OSU provider containing no information.  @throws Exception /
      [COMMENT] : Verify parcel read/write for an OSU provider containing full information.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that contained the full configuration.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that doesn't contain HomeSP.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that doesn't contain Credential.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that doesn't contain Policy.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that doesn't contain subscription update.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a configuration that doesn't contain trust root certificate list.  @throws Exception /
      [COMMENT] : Verify that a configuration without Credential is invalid.  @throws Exception /
      [COMMENT] : Verify that a configuration without HomeSP is invalid.  @throws Exception /
      [COMMENT] : Verify that a configuration with a trust root certificate URL exceeding the max size is invalid.  @throws Exception /
      [COMMENT] : Verify that a configuration with an invalid trust root certificate fingerprint is invalid.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a HomeSp containing Home Network IDs.  @throws Exception /
      [COMMENT] : Verify parcel read/write for a HomeSp without Home Network IDs.  @throws Exception /
```

[COMMENT] : Verify that a HomeSp is valid when both FQDN and Friendly Name are provided. @throws Exception /
[COMMENT] : Verify that a HomeSp is not valid when FQDN is not provided @throws Exception /
[COMMENT] : Verify that a HomeSp is not valid when Friendly Name is not provided @throws Exception /
[COMMENT] : Verify that a HomeSp is valid when the optional Home Network IDs are not provided. @throws Exception /
[COMMENT] : Verify that a HomeSp is invalid when the optional Home Network IDs contained an invalid SSID (exceeding maximum number of bytes). @throws Exception /
[COMMENT] : Verify that an user credential without CA Certificate is invalid. @throws Exception /
[COMMENT] : Verify that an user credential with EAP type other than EAP-TTLS is invalid. @throws Exception /
[COMMENT] : Verify that an user credential without realm is invalid. @throws Exception /
[COMMENT] : Verify that an user credential without username is invalid. @throws Exception /
[COMMENT] : Verify that an user credential without password is invalid. @throws Exception /
[COMMENT] : Verify that an certificate credential without CA Certificate is invalid. @throws Exception /
[COMMENT] : Verify that a certificate credential without client certificate chain is invalid. @throws Exception /
[COMMENT] : Verify that a certificate credential without client private key is invalid. @throws Exception /
[COMMENT] : Verify that a certificate credential with mismatch client certificate fingerprint is invalid. @throws Exception /
[COMMENT] : Verify that a SIM credential without IMSI is invalid. @throws Exception /
[COMMENT] : Verify that a SIM credential with an invalid IMSI is invalid. @throws Exception /
[COMMENT] : Verify that a SIM credential with invalid EAP type is invalid. @throws Exception /
[COMMENT] : Verify that a credential contained both an user and a SIM credential is invalid. @throws Exception /
[COMMENT] : Verify parcel read/write for a Policy with all fields set. @throws Exception /
[COMMENT] : Verify parcel read/write for a Policy without protocol port map. @throws Exception /
[COMMENT] : Verify parcel read/write for a Policy without preferred roaming partner list. @throws Exception /
[COMMENT] : Verify parcel read/write for a Policy without policy update parameters. @throws Exception /
[COMMENT] : Verify that a default policy (with no informatio) is invalid. @throws Exception /
[COMMENT] : Verify that a policy without policy update parameters is invalid. @throws Exception /
[COMMENT] : Verify that a policy with invalid policy update parameters is invalid. @throws Exception /
[COMMENT] : Verify that a policy with a preferred roaming partner with FQDN not specified is invalid. @throws Exception /
[COMMENT] : Verify that a policy with a preferred roaming partner with countries not specified is invalid. @throws Exception /
[COMMENT] : Verify that a policy with number of excluded SSIDs exceeded the max is invalid. @throws Exception /
[COMMENT] : Verify that a policy with an invalid SSID in the excluded SSID list is invalid. @throws Exception /
[COMMENT] : Verify parcel read/write for a UpdateParameter with all fields set. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an unknown update method is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an unknown restriction is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an username exceeding maximum size is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an empty username is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with a password exceeding maximum size is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an empty password is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with a Base64 encoded password that contained invalid padding is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter without trust root certificate URL is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with invalid trust root certificate URL is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter without trust root certificate SHA-256 fingerprint is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an incorrect size trust root certificate SHA-256 fingerprint is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter without server URI is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with an invalid server URI is invalid. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with update interval set to "never" will not perform validation on other parameters, since update is not applicable in this case. @throws Exception /
[COMMENT] : Verify that an UpdateParameter with unset update interval is invalid. @throws Exception /
[COMMENT] : Set realm for Passpoint credential; realm identifies a set of networks where your Passpoint credential can be used @param realm the realm /
[COMMENT] : Get realm for Passpoint credential; see {@link #setRealm(String)} for more information @return the realm /
[COMMENT] : Set the credential information. @param credential The credential information to set to /
[COMMENT] : Credential is needed for storing the certificates and private client key.
[COMMENT] : Set the realm associated with this credential. @param realm The realm to set to /
[COMMENT] : Get the realm associated with this credential. @return the realm associated with this credential /
[COMMENT] : Set the username associated with this user credential. @param username The username to set to /
[COMMENT] : Get the username associated with this user credential. @return the username associated with this user credential /
[COMMENT] : Set the user credential information. @param userCredential The user credential to set to /
[COMMENT] : Set the certificate type associated with this certificate credential. @param certType The certificate type to set to /

** [CLUSTER] : 11
[COMMENT] : Broadcast intent action indicating whether Wi-Fi scanning is allowed currently @hide /
[COMMENT] : Broadcast intent action indicating that Wi-Fi AP has been enabled, disabled, enabling, disabling, or failed. @hide /
[COMMENT] : Broadcast intent action indicating that the link configuration changed on wifi. @hide /
[COMMENT] : Broadcast intent action indicating that this device details have changed. /
[COMMENT] : Broadcast intent action indicating that remembered persistent groups have changed. @hide /

** [CLUSTER] : 12
[COMMENT] : Broadcast intent action indicating that a Passpoint provider icon has been received. Included extras: {@link #EXTRA_BSSID_LONG} {@link #EXTRA_FILENAME} {@link #EXTRA_ICON} Receiver Required Permission: android.Manifest.permission.ACCESS_WIFI_STATE <p>Note: The broadcast is only delivered to registered receivers - no manifest registered components will be launched. @hide /
[COMMENT] : Broadcast intent action indicating a Passpoint OSU Providers List element has been received. Included extras: {@link #EXTRA_BSSID_LONG} {@link #EXTRA_ANQP_ELEMENT_DATA} Receiver Required Permission: android.Manifest.permission.ACCESS_WIFI_STATE <p>Note: The broadcast is only delivered to registered receivers - no manifest registered components will be launched. @hide /
[COMMENT] : Broadcast intent action indicating that a Passpoint Deauth Imminent frame has been received. Included extras: {@link #EXTRA_BSSID_LONG} {@link #EXTRA_ESS} {@link #EXTRA_DELAY} {@link #EXTRA_URL} Receiver Required Permission: android.Manifest.permission.ACCESS_WIFI_STATE <p>Note: The broadcast is only delivered to registered receivers - no manifest registered components will be launched. @hide /
[COMMENT] : Broadcast intent action indicating a Passpoint subscription remediation frame has been received. Included extras: {@link #EXTRA_BSSID_LONG} {@link #EXTRA_SUBSCRIPTION_REMEDIATION_METHOD} {@link #EXTRA_URL} Receiver Required Permission: android.Manifest.permission.ACCESS_WIFI_STATE <p>Note: The broadcast is only delivered to registered receivers - no manifest registered components will be launched. @hide /

** [CLUSTER] : 13
[COMMENT] : The lookup key for an int that indicates whether Wi-Fi is enabled, disabled, enabling, disabling, or unknown. Retrieve it with {@link android.content.Intent#getIntExtra(String,int)}. @see #WIFI_STATE_DISABLED @see #WIFI_STATE_DISABLING @see #WIFI_STATE_ENABLED @see #WIFI_STATE_ENABLING @see #WIFI_STATE_UNKNOWN /
[COMMENT] : The lookup key for an int that indicates whether Wi-Fi AP is enabled, disabled, enabling, disabling, or failed. Retrieve it with {@link android.content.Intent#getIntExtra(String,int)}. @see #WIFI_AP_STATE_DISABLED @see #WIFI_AP_STATE_DISABLING @see #WIFI_AP_STATE_ENABLED @see #WIFI_AP_STATE_ENABLING @see #WIFI_AP_STATE_FAILED @hide /
[COMMENT] : Gets the Wi-Fi enabled state. @return One of {@link #WIFI_STATE_DISABLED}, {@link #WIFI_STATE_DISABLING}, {@link #WIFI_STATE_ENABLED}, {@link #WIFI_STATE_ENABLING}, {@link #WIFI_STATE_UNKNOWN} @see #isWifiEnabled() /
[COMMENT] : Gets the Wi-Fi enabled state. @return One of {@link #WIFI_AP_STATE_DISABLED}, {@link #WIFI_AP_STATE_DISABLING}, {@link #WIFI_AP_STATE_ENABLED}, {@link #WIFI_AP_STATE_ENABLING}, {@link #WIFI_AP_STATE_FAILED} @see #isWifiApEnabled() @hide /
[COMMENT] : The lookup key for an int that indicates whether Wi-Fi p2p is enabled or disabled. Retrieve it with {@link android.content.Intent#getIntExtra(String,int)}. @see #WIFI_P2P_STATE_DISABLED @see #WIFI_P2P_STATE_ENABLED /
[COMMENT] : The lookup key for an int that indicates whether p2p discovery has started or stopped. Retrieve it with {@link android.content.Intent#getIntExtra(String,int)}. @see #WIFI_P2P_DISCOVERY_STARTED @see #WIFI_P2P_DISCOVERY_STOPPED /

** [CLUSTER] : 14
[COMMENT] : Wi-Fi AP is currently being disabled. The state will change to {@link #WIFI_AP_STATE_DISABLED} if it finishes successfully. @see #WIFI_AP_STATE_CHANGED_ACTION @see #getWifiApState() @hide /
[COMMENT] : Wi-Fi AP is currently being enabled. The state will change to {@link #WIFI_AP_STATE_ENABLED} if it finishes successfully. @see #WIFI_AP_STATE_CHANGED_ACTION @see #getWifiApState() @hide /
[COMMENT] : Wi-Fi AP is in a failed state. This state will occur when an error occurs during enabling or disabling @see #WIFI_AP_STATE_CHANGED_ACTION @see #getWifiApState() @hide /
[COMMENT] : Return whether Wi-Fi AP is enabled or disabled. @return {@code true} if Wi-Fi AP is enabled @see #getWifiApState() @hide /

** [CLUSTER] : 15
	[COMMENT] : The lookup key for a {@link android.net.NetworkInfo} object associated with the Wi-Fi network. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a String giving the BSSID of the access point to which we are connected. Only present when the new state is CONNECTED. Retrieve with {@link android.content.Intent#getStringExtra(String)}. /
	[COMMENT] : The lookup key for a {@link android.net.wifi.WifiInfo} object giving the information about the access point to which we are connected. Only present when the new state is CONNECTED. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a {@link SupplicantState} describing the new state Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a {@link android.net.LinkProperties} object associated with the Wi-Fi network. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. @hide /
	[COMMENT] : The lookup key for a {@link android.net.NetworkCapabilities} object associated with the Wi-Fi network. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. @hide /
	[COMMENT] : The lookup key for a {@link android.net.wifi.p2p.WifiP2pInfo} object Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a {@link android.net.NetworkInfo} object associated with the p2p network. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a {@link android.net.wifi.p2p.WifiP2pGroup} object associated with the p2p network. Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /
	[COMMENT] : The lookup key for a {@link android.net.wifi.p2p.WifiP2pDevice} object Retrieve with {@link android.content.Intent#getParcelableExtra(String)}. /

** [CLUSTER] : 16
	[COMMENT] : A batch of access point scans has been completed and the results areavailable. Call {@link #getBatchedScanResults()} to obtain the results. @deprecated This API is nolonger supported. Use {@link android.net.wifi.WifiScanner} API @hide /
	[COMMENT] : Check if the Batched Scan feature is supported. @return false if not supported. @deprecated This API is nolonger

** [CLUSTER] : 20
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation failed due to an internal error. @hide /
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation is already in progress @hide /
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation failed due to invalid inputs @hide /
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation failed due to user permissions. @hide /
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation failed due to an internal error. /
	[COMMENT] : Passed with {@link ActionListener#onFailure}. Indicates that the operation failed because p2p is unsupported on the device. /

** [CLUSTER] : 21
	[COMMENT] : WPS operation failed @param reason The reason for failure could be one of {@link #WPS_TKIP_ONLY_PROHIBITED}, {@link #WPS_OVERLAP_ERROR}, {@link #WPS_WEP_PROHIBITED}, {@link #WPS_TIMED_OUT} or {@link #WPS_AUTH_FAILURE} and some generic errors. /
	[COMMENT] : The operation failed @param reason The reason for failure could be one of {@link #ERROR}, {@link #IN_PROGRESS} or {@link #BUSY} /
	[COMMENT] : The operation failed @param reason The reason for failure could be one of {@link #P2P_UNSUPPORTED}, {@link #ERROR} or {@link #BUSY} /

** [CLUSTER] : 22
	[COMMENT] : Connect to a network with the given configuration. The network also gets added to the list of configured networks for the foreground user. For a new network, this function is used instead of a sequence of addNetwork(), enableNetwork(), saveConfiguration() and reconnect() @param config the set of variables that describe the configuration, contained in a {@link WifiConfiguration} object. @param listener for callbacks on success or failure. Can be null. @throws IllegalStateException if the WifiManager instance needs to be initialized again @hide /
	[COMMENT] : Connect to a network with the given networkId. This function is used instead of a enableNetwork(), saveConfiguration() and reconnect() @param networkId the ID of the network as returned by {@link #addNetwork} or {@link getConfiguredNetworks}. @param listener for callbacks on success or failure. Can be null. @throws IllegalStateException if the WifiManager instance needs to be initialized again @hide /
	[COMMENT] : Save the given network to the list of configured networks for the foreground user. If the network already exists, the configuration is updated. Any new network is enabled by default. For a new network, this function is used instead of a sequence of addNetwork(), enableNetwork() and saveConfiguration(). For an existing network, it accomplishes the task of updateNetwork() and saveConfiguration() @param config the set of variables that describe the configuration, contained in a {@link WifiConfiguration} object. @param listener for callbacks on success or failure. Can be null. @throws IllegalStateException if the WifiManager instance needs to be initialized again @hide /
	[COMMENT] : Delete the network from the list of configured networks for the foreground user. This function is used instead of a sequence of removeNetwork() and saveConfiguration(). @param config the set of variables that describe the configuration, contained in a {@link WifiConfiguration} object. @param listener for callbacks on success or failure. Can be null. @throws IllegalStateException if the WifiManager instance needs to be initialized again @hide /

** [CLUSTER] : 23
	[COMMENT] : Get a reference to WifiService handler. This is used by a client to establish an AsyncChannel communication with WifiService @return Messenger pointing to the WifiService handler @hide /
	[COMMENT] : Get a reference to WifiP2pService handler. This is used to establish an AsyncChannel communication with WifiService @param binder A binder for the service to associate with this client. @return Messenger pointing to the WifiP2pService handler @hide /
	[COMMENT] : Get a reference to P2pStateMachine handler. This is used to establish a priveleged AsyncChannel communication with WifiP2pService. @return Messenger pointing to the WifiP2pService handler @hide /

** [CLUSTER] : 24
	[COMMENT] : Locks the Wi-Fi radio on until {@link #release} is called. If this WifiLock is reference-counted, each call to {@code acquire} will increment the reference count, and the radio will remain locked as long as the reference count is above zero. If this WifiLock is not reference-counted, the first call to {@code acquire} will lock the radio, but subsequent calls will be ignored. Only one call to {@link #release} will be required, regardless of the number of times that {@code acquire} is called. /
	[COMMENT] : Unlocks the Wi-Fi radio, allowing it to turn off when the device is idle. If this WifiLock is reference-counted, each call to {@code release} will decrement the reference count, and the radio will be unlocked only when the reference count reaches zero. If the reference count goes below zero (that is, if {@code release} is called a greater number of times than {@link #acquire}), an exception is thrown. If this WifiLock is not reference-counted, the first call to {@code release} (after the radio was locked using {@link #acquire}) will unlock the radio, and subsequent calls will be ignored. /
	[COMMENT] : Locks Wifi Multicast on until {@link #release} is called. If this MulticastLock is reference-counted each call to {@code acquire} will increment the reference count, and the wifi interface will receive multicast packets as long as the reference count is above zero. If this MulticastLock is not reference-counted, the first call to {@code acquire} will turn on the multicast packets, but subsequent calls will be ignored. Only one call to {@link #release} will be required, regardless of the number of times that {@code acquire} is called. Note that other applications may also lock Wifi Multicast on. Only they can relinquish their lock. Also note that applications cannot leave Multicast locked on. When an app exits or crashes, any Multicast locks will be released. /
	[COMMENT] : Unlocks Wifi Multicast, restoring the filter of packets not addressed specifically to this device and saving power. If this MulticastLock is reference-counted, each call to {@code release} will decrement the reference count, and the multicast packets will only stop being received when the reference count reaches zero. If the reference count goes below zero (that is, if {@code release} is called a greater number of times than {@link #acquire}), an exception is thrown. If this MulticastLock is not reference-counted, the first call to {@code release} (after the radio was multicast locked using {@link #acquire}) will unlock the multicast, and subsequent calls will be ignored. Note that if any other Wifi Multicast Locks are still outstanding this {@code release} call will not have an immediate effect. Only when all applications have released all their Multicast Locks will the Multicast filter be turned back on. Also note that when an app exits or crashes all of its Multicast Locks will be automatically released. /

** [CLUSTER] : 25
	[COMMENT] : Creates a new WifiLock. @param lockType the type of lock to create. See {@link #WIFI_MODE_FULL}, {@link #WIFI_MODE_FULL_HIGH_PERF} and {@link #WIFI_MODE_SCAN_ONLY} for descriptions of the types of Wi-Fi locks. @param tag a tag for the WifiLock to identify it in debugging messages. This string is never shown to the user under normal conditions, but should be descriptive enough to identify your application and the specific WifiLock within it, if it holds multiple WifiLocks. @return a new, unacquired

** [CLUSTER] : 29
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_20_SUPPORT} API. /
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_40_SUPPORT} API. /
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_80_SUPPORT} API. /
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_160_SUPPORT} API. /
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_5_SUPPORT} API. /
	[COMMENT] : @deprecated It is not supported anymore. Use {@link android.net.wifi.RttManager#RTT_BW_10_SUPPORT} API. /

** [CLUSTER] : 30
	[COMMENT] : Not used if the AP bandwidth is 20 MHz If the AP use 40, 80 or 160 MHz, this is the center frequency if the AP use 80 + 80 MHz, this is the center frequency of the first segment same as ScanResult.centerFreq0 Default value: 0 /
	[COMMENT] : Only used if the AP bandwidth is 80 + 80 MHz if the AP use 80 + 80 MHz, this is the center frequency of the second segment same as ScanResult.centerFreq1 Default value: 0 /
	[COMMENT] : Center frequency of the second segment when channel width is 80 + 80 MHz. @see ScanResult#centerFreq1 /
	[COMMENT] : AP Channel bandwidth is 160 MHZ, but 80MHZ + 80MHZ /
	[COMMENT] : AP Channel bandwidth; one of {@link #CHANNEL_WIDTH_20MHZ}, {@link #CHANNEL_WIDTH_40MHZ}, {@link #CHANNEL_WIDTH_80MHZ}, {@link #CHANNEL_WIDTH_160MHZ} or {@link #CHANNEL_WIDTH_80MHZ_PLUS_MHZ}. /
	[COMMENT] : Not used if the AP bandwidth is 20 MHz If the AP use 40, 80 or 160 MHz, this is the center frequency (in MHz) if the AP use 80 + 80 MHz, this is the center frequency of the first segment (in MHz) /
	[COMMENT] : Only used if the AP bandwidth is 80 + 80 MHz if the AP use 80 + 80 MHz, this is the center frequency of the second segment (in MHz) /

** [CLUSTER] : 31
	[COMMENT] : RSSI spread (i.e. max - min) @deprecated Use {@link android.net.wifi.RttManager.RttResult#rssiSpread} API. /
	[COMMENT] : spread (i.e. max - min) round trip time @deprecated Use {@link android.net.wifi.RttManager.RttResult#rttSpread} API. /
	[COMMENT] : spread (i.e. max - min) distance @deprecated Use {@link android.net.wifi.RttManager.RttResult#distanceSpread} API. /

** [CLUSTER] : 42
	[COMMENT] : The current status of this network configuration entry. @see Status /
	[COMMENT] : Quality network selection status String (for debug purpose). Use Quality network selection status value as index to extec the corresponding debug string /
	[COMMENT] : get current Quality network selection status @return returns current Quality network selection status in String (for debug purpose) /
	[COMMENT] : @param reason specific error reason @return corresponding network disable reason String (for debug purpose) /
	[COMMENT] : get current network disable reason @return current network disable reason in String (for debug purpose) /
	[COMMENT] : get current network network selection status @return return current network network selection status /
	[COMMENT] : set current networ work selection status @param status network selection status to set /
	[COMMENT] : BSSID for connection to this network (through network selection procedure) /
	[COMMENT] : get current network Selection BSSID @return current network Selection BSSID /
	[COMMENT] : set network Selection BSSID @param bssid The target BSSID for assocaition /

** [CLUSTER] : 37
	[COMMENT] : The network's SSID. Can either be an ASCII string, which must be enclosed in double quotation marks (e.g., {@code "MyNetwork"}, or a string of hex digits,which are not enclosed in quotes (e.g., {@code 01a243f405}). /
	[COMMENT] : The network's SSID. Can either be an ASCII string, which must be enclosed in double quotation marks (e.g., {@code "MyNetwork"}), or a string of hex digits, which are not enclosed in quotes (e.g., {@code 01a243f405}). /
	[COMMENT] : Pre-shared key for use with WPA-PSK. Either an ASCII string enclosed in double quotation marks (e.g., {@code "abcdefghij"} for PSK passphrase or a string of 64 hex digits for raw PSK. When the value of this key is read, the actual key is not returned, just a "*" if the key has a value, or the null string otherwise. /
	[COMMENT] : Up to four WEP keys. Either an ASCII string enclosed in double quotation marks (e.g., {@code "abcdef"}) or a string of hex digits (e.g., {@code 0102030405}). When the value of one of these keys is read, the actual key is not returned, just a "*" if the key has a value, or the null string otherwise. /

** [CLUSTER] : 32
	[COMMENT] : WME Best Effort Access Category (receive mpdu, transmit mpdu, lost mpdu, number of retries)
	[COMMENT] : WME Background Access Category (receive mpdu, transmit mpdu, lost mpdu, number of retries)
	[COMMENT] : WME Video Access Category (receive mpdu, transmit mpdu, lost mpdu, number of retries)
	[COMMENT] : WME Voice Access Category (receive mpdu, transmit mpdu, lost mpdu, number of retries)

** [CLUSTER] : 33
	[COMMENT] : Encode a CA certificate alias so it does not contain illegal character. @hide /
	[COMMENT] : Set CA certificate alias. See the {@link android.security.KeyChain} for details on installing or choosing a certificate @param alias identifies the certificate @hide /
	[COMMENT] : Set CA certificate aliases. When creating installing the corresponding certificate to the keystore, please use alias encoded by {@link #encodeCaCertificateAlias(String)}. See the {@link android.security.KeyChain} for details on installing or choosing a certificate. @param aliases identifies the certificate @hide /
	[COMMENT] : Get CA certificate alias @return alias to the CA certificate @hide /
	[COMMENT] : Get CA certificate aliases @return alias to the CA certificate @hide /
	[COMMENT] : Set Client certificate alias. See the {@link android.security.KeyChain} for details on installing or choosing a certificate @param alias identifies the certificate @hide /
	[COMMENT] : Get client certificate alias @return alias to the client certificate @hide /

** [CLUSTER] : 115
	[COMMENT] : Represents the notification parameters that can be included in a {@link Message}. Instances of this class are thread-safe and immutable. /
	[COMMENT] : Represents the Android-specific options that can be included in a {@link Message}. Instances of this class are thread-safe and immutable. /
	[COMMENT] : Represents the Webpush-specific notification options that can be included in a {@link Message}. Instances of this class are thread-safe and immutable. /
	[COMMENT] : Represents the Webpush protocol options that can be included in a {@link Message}. Instances of this class are thread-safe and immutable. /
	[COMMENT] : Represents the Android-specific notification options that can be included in a {@link Message}. Instances of this class are thread-safe and immutable. /

** [CLUSTER] : 116
	[COMMENT] : Creates a new {@code Notification} using the given title and body. @param title Title of the notification. @param body Body of the notification. /
	[COMMENT] : Creates a new notification with the given title and body. Overrides the options set via {@link Notification}. @param title Title of the notification. @param body Body of the notification. /
	[COMMENT] : Creates a new notification with the given title, body and icon. Overrides the options set via {@link Notification}. @param title Title of the notification. @param body Body of the notification. @param icon URL to the notifications icon. /

** [CLUSTER] : 117
	[COMMENT] : Sets the title of the alert. When provided, overrides the title sent via {@link Notification}. @param title Title of the notification. @return This builder. /
	[COMMENT] : Sets the body of the alert. When provided, overrides the body sent via {@link Notification}. @param body Body of the notification. @return This builder. /
	[COMMENT] : Sets the title of the Android notification. When provided, overrides the title set via {@link Notification}. @param title Title of the notification. @return This builder. /
	[COMMENT] : Sets the body of the Android notification. When provided, overrides the body sent via {@link Notification}. @param body Body of the notification. @return This builder. /

** [CLUSTER] : 118
	[COMMENT] : Sets the key of the text in the app's string resources to use to localize the action button text. @param actionLocKey Resource key string. @return This builder. /
	[COMMENT] : Sets the key of the body string in the app's string resources to use to localize the body text. @param locKey Resource key string. @return This builder. /
	[COMMENT] : Sets the key of the title string in the app's string resources to use to localize the title text. @param titleLocKey Resource key string. @return This builder. /
	[COMMENT] : Sets the key of the body string in the app's string resources to use to localize the body text. @param bodyLocKey Resource key string. @return This builder. /

** [CLUSTER] : 119
	[COMMENT] : Adds a resource key string that will be used in place of the format specifiers in {@code bodyLocKey}. @param arg Resource key string. @return This builder. /
	[COMMENT] : Adds a list of resource keys that will be used in place of the format specifiers in {@code bodyLocKey}. @param args List of resource key strings. @return This builder. /
	[COMMENT] : Adds a resource key string that will be used in place of the format specifiers in {@code titleLocKey}. @param arg Resource key string. @return This builder. /
	[COMMENT] : Adds a list of resource keys that will be used in place of the format specifiers in {@code titleLocKey}. @param args List of resource key strings. @return This builder. /

** [CLUSTER] : 120
	[COMMENT] : Retrieves an {@link android.mtp.MtpDevice} object for the USB device with the given name. @param deviceName the name of the USB device @return the MtpDevice, or null if it does not exist /
	[COMMENT] : Retrieves an {@link android.mtp.MtpDevice} object for the USB device with the given ID. @param id the ID of the USB device @return the MtpDevice, or null if it does not exist /
	[COMMENT] : Retrieves a list of all currently connected {@link android.mtp.MtpDevice}. @return the list of MtpDevices /

** [CLUSTER] : 121
	[COMMENT] : Retrieves a list of all {@link android.mtp.MtpStorageInfo} for the MTP or PTP device with the given USB device name @param deviceName the name of the USB device @return the list of MtpStorageInfo /
	[COMMENT] : Retrieves the {@link android.mtp.MtpObjectInfo} for an object on the MTP or PTP device with the given USB device name with the given object handle @param deviceName the name of the USB device @param objectHandle handle of the object to query @return the MtpObjectInfo /
	[COMMENT] : Deletes an object on the MTP or PTP device with the given USB device name. @param deviceName the name of the USB device @param objectHandle handle of the object to delete @return true if the deletion succeeds /
	[COMMENT] : Retrieves a list of {@link android.mtp.MtpObjectInfo} for all objects on the MTP or PTP device with the given USB device name and given storage ID and/or object handle. If the object handle is zero, then all objects in the root of the storage unit will be returned. Otherwise, all immediate children of the object will be returned. If the storage ID is also zero, then all objects on all storage units will be returned. @param deviceName the name of the USB device @param storageId the ID of the storage unit to query, or zero for all @param objectHandle the handle of the parent object to query, or zero for the storage root @return the list of MtpObjectInfo /

** [CLUSTER] : 163
	[COMMENT] : Returns the list of object handles for all objects on the given storage unit, with the given format and parent. Information about each object can be accessed via {@link #getObjectInfo}. @param storageId the storage unit to query @param format the format of the object to return, or zero for all formats @param objectHandle the parent object to query, -1 for the storage root, or zero for all objects @return the object handles, or null if fetching object handles fails /
	[COMMENT] : Retrieves the object handle for the parent of an object on the device. @param objectHandle handle of the object to query @return the parent's handle, or zero if it is in the root of the storage /
	[COMMENT] : Retrieves the ID of the storage unit containing the given object on the device. @param objectHandle handle of the object to query @return the object's storage unit ID /
	[COMMENT] : Returns the object handle for the object's parent Will be zero for the root directory of a storage unit @return the object's parent /

** [CLUSTER] : 165
	[COMMENT] : This class encapsulates information about a storage unit on an MTP device. This corresponds to the StorageInfo Dataset described in section 5.2.2 of the MTP specification. /
	[COMMENT] : This class encapsulates information about an object on an MTP device. This corresponds to the ObjectInfo Dataset described in section 5.3.1 of the MTP specification. /
	[COMMENT] : This class encapsulates information about an MTP device. This corresponds to the DeviceInfo Dataset described in section 5.1.1 of the MTP specification. /

** [CLUSTER] : 166
	[COMMENT] : Returns the storage ID for the storage unit. The storage ID uniquely identifies the storage unit on the MTP device. @return the storage ID /
	[COMMENT] : Returns the description string for the storage unit. This is typically displayed to the user in the user interface on the MTP host. @return the storage unit description /
	[COMMENT] : Returns the volume identifier for the storage unit @return the storage volume identifier /
	[COMMENT] : Returns the object handle for the MTP object @return the object handle /
	[COMMENT] : Returns the storage ID for the MTP object's storage unit @return the storage ID /
	[COMMENT] : Returns the format code for the MTP object @return the format code /
	[COMMENT] : Returns the size of the MTP object @return the object size /
	[COMMENT] : Returns the format code for the MTP object's thumbnail Will be zero for objects with no thumbnail @return the thumbnail format code /
	[COMMENT] : Returns the size of the MTP object's thumbnail Will be zero for objects with no thumbnail @return the thumbnail size /
	[COMMENT] : Returns the width of the MTP object's thumbnail in pixels Will be zero for objects with no thumbnail @return the thumbnail width /
	[COMMENT] : Returns the height of the MTP object's thumbnail in pixels Will be zero for objects with no thumbnail @return the thumbnail height /
	[COMMENT] : Returns the width of the MTP object in pixels Will be zero for non-image objects @return the image width /
	[COMMENT] : Returns the height of the MTP object in pixels Will be zero for non-image objects @return the image height /
	[COMMENT] : Returns the depth of the MTP object in bits per pixel Will be zero for non-image objects @return the image depth /
	[COMMENT] : Returns the name of the MTP object @return the object's name /
	[COMMENT] : Returns the storage ID for the storage unit @return the storage ID /
	[COMMENT] : Returns the file path for the storage unit's storage in the file system @return the storage file path /
	[COMMENT] : Returns the description string for the storage unit @return the storage unit description /

** [CLUSTER] : 167
	[COMMENT] : Object is not protected. It may be modified and deleted, and its properties may be modified. /
	[COMMENT] : Object can not be modified or deleted and its properties can not be modified. /
	[COMMENT] : Object can not be modified or deleted but its properties are modifiable. /
	[COMMENT] : Object's contents can not be transfered from the device, but the object may be moved or deleted and its properties may be modified. /

** [CLUSTER] : 168
	[COMMENT] : The {@link Intent} that must be declared as handled by the service. /
	[COMMENT] : The {@link Intent} that must be declared as handled by the service. The service must also require the {@link android.Manifest.permission#BIND_TV_REMOTE_SERVICE} permission so that other applications cannot abuse it. /
	[COMMENT] : The {@link Intent} that must be declared as handled by the service. Put this in your manifest. /