Filtering Unwanted Messages with Identity Lookup

Session 249

Stuart Montgomery, iOS Engineer
Stuart Montgomery
Hey what’s going on?
Stuart Montgomery
Hey what's going on?

5/17/17
Stuart Montgomery

Hey what’s going on?

5/17/17
Stuart Montgomery
Hey what’s going on?
5/17/17
Messages

Search

iMessage & SMS  SMS Junk (1)

+1 (408) 555-1234  5:42 PM
You've been selected to win a FREE $1...
Messages

Search

iMessage & SMS  SMS Junk (1)

+1 (408) 555-1234  5:42 PM
You've been selected to win a FREE $1...
Motivation
Motivation

Increasing nuisance for users
Motivation

Increasing nuisance for users

Sometimes phishing attempts
Motivation

Increasing nuisance for users

Sometimes phishing attempts

Delivered via carrier, unlike iMessage
Motivation

Increasing nuisance for users

Sometimes phishing attempts

Delivered via carrier, unlike iMessage

Leverage expertise in detecting unwanted messages
Message Filter extensions
Privacy considerations
Network deferral
Demo
Message Filter extensions
Privacy considerations
Network deferral
Demo
Message Filter extensions
Privacy considerations
Network deferral
Demo
Message Filter extensions
Privacy considerations
Network deferral
Demo
Message Filter extensions
Privacy considerations
Network deferral
Demo
Message Filter Extensions
Message Filter Extensions

New app extension point
Message Filter Extensions

New app extension point

Part of Identity Lookup framework, new in iOS 11
Message Filter Extensions

New app extension point

Part of Identity Lookup framework, new in iOS 11

Must be enabled by user in Messages settings
Message Filter Extensions

New app extension point

Part of Identity Lookup framework, new in iOS 11

Must be enabled by user in Messages settings

One extension enabled at a time
Message Filter Extensions

New app extension point

Part of Identity Lookup framework, new in iOS 11

Must be enabled by user in Messages settings

One extension enabled at a time

Invoked for each SMS from an unknown sender
Message Filter Extensions

Offline-only
Message Filter Extensions

Offline-only

Junk

Messages

MessageFilter Extension

App
Message Filter Extensions

Offline-only
Message Filter Extensions

Offline-only

[Diagram showing relationships between Messages, ILMMessageFilter, QueryRequest, MessageFilter Extension, and App]
Message Filter Extensions

Offline-only

ILMessageFilter
QueryRequest

ILMessageFilter
QueryResponse

MessageFilter
Extension

App
Privacy Considerations
Privacy Considerations
Privacy Considerations

Recipient’s phone number not sent to extension, only sender’s
Privacy Considerations

Recipient’s phone number not sent to extension, only sender’s

Restrictions on extension
Privacy Considerations

Recipient’s phone number not sent to extension, only sender’s

Restrictions on extension

• Cannot write to files shared with containing app
Privacy Considerations

Recipient’s phone number not sent to extension, only sender’s

Restrictions on extension
• Cannot write to files shared with containing app
• Cannot perform networking
Privacy Considerations

Recipient’s phone number not sent to extension, only sender’s

Restrictions on extension
• Cannot write to files shared with containing app
• Cannot perform networking

Do not export messages outside extension’s container
Message Criteria
Requirements for messages to be sent to extension
Message Criteria
Requirements for messages to be sent to extension

SMS and MMS only
Message Criteria
Requirements for messages to be sent to extension

SMS and MMS only
Never iMessage
Message Criteria
Requirements for messages to be sent to extension

SMS and MMS only

Never iMessage

Only unknown senders (not in recipient’s Contacts)
Message Criteria
Requirements for messages to be sent to extension

SMS and MMS only

Never iMessage

Only unknown senders (not in recipient’s Contacts)

After multiple responses
Message Criteria
Requirements for messages to be sent to extension

SMS and MMS only

Never iMessage

Only unknown senders (not in recipient’s Contacts)

After multiple responses
  • Stops sending to extension
Message Criteria
Requirements for messages to be sent to extension

SMS and MMS only

Never iMessage

Only unknown senders (not in recipient’s Contacts)

After multiple responses
  • Stops sending to extension
  • Junk threads promoted to non-junk
Demo
Offline-only extension
Deployment Target

iOS Deployment Target: 11.0

Configurations

- Name: Development
  - Based on Configuration File
  - No Configurations Set

Localizations

- Language: English
  - Development Language
  - 2 Files Localized

- Use Base Internationalization
final class MessageFilterExtension: ILMessageFilterExtension {}

extension MessageFilterExtension: ILMessageFilterQueryHandling {

    func handle(_ queryRequest: ILMessageFilterQueryRequest, context: ILMessageFilterExtensionContext, completion: @escaping (ILMessageFilterQueryResponse) -> Void) {
        // First, check whether to filter using offline data (if possible).
        let offlineAction = self.offlineAction(for: queryRequest)

        switch offlineAction {
        case .allow, .filter:
            // Based on offline data, we know this message should either be Allowed or Filtered. Send response immediately.
            let response = ILMessageFilterQueryResponse()
            response.action = offlineAction
            completion(response)

        case .none:
            // Based on offline data, we do not know whether this message should be Allowed or Filtered. Defer to network.
            // Note: Deferring requests to network requires the extension target's Info.plist to contain a key with a URL to use. See documentation for details.
            context.deferQueryRequestToNetwork() {
                (networkResponse, error) in
                let response = ILMessageFilterQueryResponse()
                response.action = .none
            }
        }
    }
}
final class MessageFilterExtension: ILMessageFilterExtension {}

extension MessageFilterExtension: ILMessageFilterQueryHandling {

    func handle(_ queryRequest: ILMessageFilterQueryRequest, context: ILMessageFilterExtensionContext, completion: @escaping (ILMessageFilterQueryResponse) -> Void) {
        // First, check whether to filter using offline data (if possible).
        let offlineAction = self.offlineAction(for: queryRequest)

        switch offlineAction {
        case .allow, .filter:
            // Based on offline data, we know this message should either be Allowed or Filtered. Send response immediately.
            let response = ILMessageFilterQueryResponse()
            response.action = offlineAction

            completion(response)

        case .none:
            // Based on offline data, we do not know whether this message should be Allowed or Filtered. Defers to network.
            // Note: Deferring requests to network requires the extension target's Info.plist to contain a key with a URL to use. See documentation for details.
                context.deferQueryRequestToNetwork() { (networkResponse, error) in
                let response = ILMessageFilterQueryResponse()
                response.action = .none
            }
final class MessageFilterExtension: ILMMessageFilterExtension {}

extension MessageFilterExtension: ILMMessageFilterQueryHandling {

    func handle(_ queryRequest: ILMMessageFilterQueryRequest, context: ILMMessageFilterExtensionContext, completion: @escaping (ILMessageFilterQueryResponse) -> Void) {
        // First, check whether to filter using offline data (if possible).
        let offlineAction = self.offlineAction(for: queryRequest)

        switch offlineAction {
        case .allow, .filter:
            // Based on offline data, we know this message should either be Allowed or Filtered. Send response immediately.
            let response = ILMMessageFilterQueryResponse()
            response.action = offlineAction

            completion(response)

        case .none:
            // Based on offline data, we do not know whether this message should be Allowed or Filtered. Defer to network.
            // Note: Deferring requests to network requires the extension target's Info.plist to contain a key with a URL to use. See documentation for details.
            context.deferQueryRequestToNetwork() { (networkResponse, error) in
                let response = ILMMessageFilterQueryResponse()
                response.action = .none
            }
        }
    }
}
final class MessageFilterExtension: ILMMessageFilterExtension {}

extension MessageFilterExtension: ILMMessageFilterQueryHandling {

    func handle(_ queryRequest: ILMMessageFilterQueryRequest, context: ILMMessageFilterExtensionContext, completion: @escaping (ILMessageFilterQueryResponse) ->Void) {
        // First, check whether to filter using offline data (if possible).
        let offlineAction = self.offlineAction(for: queryRequest)

        switch offlineAction {
            case .allow, .filter:
                // Based on offline data, we know this message should either be Allowed or Filtered. Send response immediately.
                let response = ILMMessageFilterQueryResponse()
                response.action = offlineAction

                completion(response)

            case .none:
                // Based on offline data, we do not know whether this message should be Allowed or Filtered. Defer to network.
                // Note: Deferring requests to network requires the extension target's Info.plist to contain a key with a URL to use. See documentation for details.
                context.deferQueryRequestToNetwork() {
                    (networkResponse, error) in
                    let response = ILMMessageFilterQueryResponse()
                    response.action = .none
                    completion(response)
                }
        }
    }
}
final class IMessageFilterExtension: ILMessageFilterExtension {}

extension IMessageFilterExtension: ILMessageFilterQueryHandling {

    func handle(_ queryRequest: ILMessageFilterQueryRequest, context: ILMessageFilterExtensionContext, completion: @escaping (ILMessageFilterQueryResponse) -> Void) {
        // First, check whether to filter using offline data (if possible).
        let offlineAction = self.offlineAction(for: queryRequest)

        switch offlineAction {
        case .allow, .filter:
            // Based on offline data, we know this message should either be Allowed or Filtered. Send response immediately.
            let response = ILMessageFilterQueryResponse()
            response.action = offlineAction

            completion(response)

        case .none:
            // Based on offline data, we do not know whether this message should be Allowed or Filtered. Deferral to network.
            // Note: Deferring requests to network requires the extension target's Info.plist to contain key with a URL to use. See documentation for details.
            context.deferQueryRequestToNetwork() { (networkResponse, error) in
                let response = ILMessageFilterQueryResponse()
                response.action = .none
            }
        }
    }
}
final class MessageFilterExtension: ILMessageFilterExtension {}

extension MessageFilterExtension: ILMessageFilterQueryHandling {

    func handle(_ queryRequest: ILMessageFilterQueryRequest, context: ILMessageFilterExtensionContext, completion: @escaping (ILMessageFilterQueryResponse) -> Void) {
        // First, check whether to filter using offline data (if possible).
        let offlineAction = self.offlineAction(for: queryRequest)

        switch offlineAction {
        case .allow, .filter:
            // Based on offline data, we know this message should either be Allowed or Filtered. Send response immediately.
            let response = ILMessageFilterQueryResponse()
            response.action = offlineAction

            completion(response)

        case .none:
            // Based on offline data, we do not know whether this message should be Allowed or Filtered. Defer to network.
            // Note: Deferring requests to network requires the extension target's Info.plist to contain a key with a URL to use. See documentation for details.
            context.deferQueryRequestToNetwork() { (networkResponse, error) in
                let response = ILMessageFilterQueryResponse()
                response.action = .none

                completion(response)
final class MessageFilterExtension: ILMMessageFilterExtension {}

extension MessageFilterExtension: ILMMessageFilterQueryHandling {

    func handle(_ queryRequest: ILMMessageFilterQueryRequest, context: ILMMessageFilterExtensionContext, completion: @escaping (ILMessageFilterQueryResponse) -> Void) {
        // First, check whether to filter using offline data (if possible).
        let offlineAction = self.offlineAction(for: queryRequest)

        switch offlineAction {
            case .allow, .filter:
                // Based on offline data, we know this message should either be Allowed or Filtered. Send response immediately.
                let response = ILMMessageFilterQueryResponse()
                response.action = offlineAction
                completion(response)

            case .none:
                // Based on offline data, we do not know whether this message should be Allowed or Filtered. Defer to network.
                // Note: Deferring requests to network requires the extension target's Info.plist to contain a key with a URL to use. See documentation for details.
                context.deferQueryRequestToNetwork() {
                    (networkResponse, error) in
                    let response = ILMMessageFilterQueryResponse()
                    response.action = .none
                    completion(response)
                }
        }
    }
}
final class MessageFilterExtension: ILMessageFilterExtension {

extension MessageFilterExtension: ILMessageFilterQueryHandling {

    func handle(_ queryRequest: ILMessageFilterQueryRequest, context: ILMessageFilterExtensionContext, completion: @escaping (ILMessageFilterQueryResponse) -> Void) {
        // First, check whether to filter using offline data (if possible).
        let offlineAction = self.offlineAction(for: queryRequest)

        switch offlineAction {
        case .allow, .filter:
            // Based on offline data, we know this message should either be Allowed or Filtered. Send response immediately.
            let response = ILMessageFilterQueryResponse()
            response.action = offlineAction
            completion(response)

        case .none:
            // Based on offline data, we do not know whether this message should be Allowed or Filtered. Defer to network.
            // Note: Deferring requests to network requires the extension target's Info.plist to contain a key with a URL to use. See documentation for details.
            context.deferQueryRequestToNetwork() {
                (networkResponse, error) in
                let response = ILMessageFilterQueryResponse()
                response.action = .none
            }
        }
    }
}
if let networkResponse = networkResponse {
    // If we received a network response, parse it to determine an action
to return in our response.
    response.action = self.action(for: networkResponse)
} else {
    NSLog("Error deferring query request to network: \(String(describing: error))")
}

completion(response)
}

private func offlineAction(for queryRequest: IMessageFilterQueryRequest) -> IMessageFilterAction {
    // Replace with logic to perform offline check whether to filter first (if possible).
    return .none
}

private func action(for networkResponse: INetworkResponse) -> IMessageFilterAction {
    // Replace with logic to parse the HTTP response and data payload of
    // networkResponse to return an action.
    return .none
}
if let networkResponse = networkResponse {
    // If we received a network response, parse it to determine an action to return in our response.
    response.action = self.action(for: networkResponse)
} else {
    NSLog("Error deferring query request to network: \(String(describing: error))")
}

completion(response)

private func offlineAction(for queryRequest: ILMMessageFilterQueryRequest) -> ILMMessageFilterAction {
    // Replace with logic to perform offline check whether to filter first (if possible).
    return .none
}

private func action(for networkResponse: ILMNetworkResponse) -> ILMMessageFilterAction {
    // Replace with logic to parse the HTTP response and data payload of `networkResponse` to return an action.
    return .none
}
Network Deferral
Message Filter Extensions
With network deferral

App
MessageFilter Extension

App Web Server
Message Filter Extensions
With network deferral

App
MessageFilter Extension

ILMessageFilterQueryRequest

App Web Server
Message Filter Extensions
With network deferral

App
MessageFilter Extension

ILMessageFilterQueryRequest

Defer to network

App Web Server
Message Filter Extensions
With network deferral

Diagram:
- App
  - MessageFilter Extension
- App Web Server
  - JSON Request

Flow:
- ILMessageFilterQueryRequest
- Defer to network
- JSON Request
Message Filter Extensions
With network deferral

App
MessageFilter Extension

Server Response

App Web Server

ILMessageFilterQueryRequest
Defer to network
JSON Request

Server Response
Message Filter Extensions
With network deferral

App
MessageFilter Extension

Defer to network
JSON Request
Server Response

App Web Server
Network Deferral Restrictions
Network Deferral Restrictions

Network requests contain no personally identifiable information
Network Deferral Restrictions

Network requests contain no personally identifiable information.

URL specified in extension’s Info.plist:

```
NSExtension

NSExtensionAttributes

ILMessageFilterExtensionNetworkURL: https://www.example.com/my-message-filter-server
```
Network Deferral Restrictions

Network requests contain no personally identifiable information

URL specified in extension's Info.plist
Network Deferral Restrictions

Network requests contain no personally identifiable information

URL specified in extension’s Info.plist

HTTPS required, no App Transport Security (ATS) overrides
Network Deferral Restrictions

Network requests contain no personally identifiable information
URL specified in extension’s Info.plist
HTTPS required, no App Transport Security (ATS) overrides
Extension’s app must use Associated Domains (apple-app-site-association)
Network Deferral Restrictions

Network requests contain no personally identifiable information

URL specified in extension’s Info.plist

HTTPS required, no App Transport Security (ATS) overrides

Extension’s app must use Associated Domains (apple-app-site-association)
Network Deferral Restrictions

Network requests contain no personally identifiable information

URL specified in extension’s Info.plist

HTTPS required, no App Transport Security (ATS) overrides

Extension’s app must use Associated Domains (apple-app-site-association)

Cookies are not preserved
Network Request and Response Formats
Network Request and Response Formats

Network request uses JSON and includes
Network Request and Response Formats

Network request uses JSON and includes
• Message sender (phone number or email)
Network Request and Response Formats

Network request uses JSON and includes
- Message sender (phone number or email)
- Message body
Network Request and Response Formats

Network request uses JSON and includes

• Message sender (phone number or email)
• Message body
• Version of app (CFBundleVersion)
Network Request and Response Formats

Network request uses JSON and includes:

- Message sender (phone number or email)
- Message body
- Version of app (CFBundleVersion)
- Version of the JSON request format (currently 1)
Network Request and Response Formats

Network request uses JSON and includes

• Message sender (phone number or email)
• Message body
• Version of app (CFBundleVersion)
• Version of the JSON request format (currently 1)

Response format is up to your app to define.
// Example of Network HTTP Request Sent to Server

POST /server-url-path HTTP/1.1
Accept: */*
Content-Type: application/json; charset=utf-8
Content-Length: 148

{   "_version": 1,
    "query": {
        "sender": "14085550001",
        "message": {
            "text": "This is a message"
        }
    },
    "app": {
        "version": "1.1"
    }
}
Demo
Network-deferring extension
func handle(_ queryRequest: ILMessageFilterQueryRequest, context: ILMessageFilterExtensionContext, completion: @escaping (ILMessageFilterQueryResponse) -> Void) {

    // First, check whether to filter offline data (if possible).
    let offlineAction = self.offlineAction(for: queryRequest)

    switch offlineAction {
    case .allow, .filter:
        // Based on offline data, we know this message should either be Allowed or Filtered. Send response immediately.
        let response = ILMessageFilterQueryResponse()
        response.action = offlineAction
        completion(response)

    case .none:
        // Based on offline data, we do not know whether this message should be Allowed or Filtered. Defers to network.
        // Note: Deferring requests to network requires the extension target's Info.plist to contain a key with a URL to use. See documentation for details.
        context.deferQueryRequestToNetwork() { (networkResponse, error) in
            let response = ILMessageFilterQueryResponse()
            response.action = .none

            if let networkResponse = networkResponse {
                // If we received a network response, parse it to determine an action to return in our response.
                response.action = self.action(for: networkResponse)
            }

            completion(response)
        }
    }
}
```swift
func handle(_ queryRequest: ILMMessageFilterQueryRequest, context: ILMMessageFilterExtensionContext, completion: @escaping (ILMessageFilterQueryResponse) -> Void) {

    // First, check whether to filter offline data (if possible).
    let offlineAction = self.offlineAction(for: queryRequest)

    switch offlineAction {
    case .allow, .filter:
        // Based on offline data, we know this message should either be Allowed or Filtered. Send response immediately.
        let response = ILMMessageFilterQueryResponse()
        response.action = offlineAction

        completion(response)

    case .none:
        // Based on offline data, we do not know whether this message should be Allowed or Filtered. Defer to network.
        // Note: Deferring requests to network requires the extension target's Info.plist to contain a key with a URL to use. See documentation for details.

        context.deferQueryRequestToNetwork() { (networkResponse, error) in
            let response = ILMMessageFilterQueryResponse()
            response.action = .none

            if let networkResponse = networkResponse {
                // If we received a network response, parse it to determine an action to return in our response.
                response.action = self.action(for: networkResponse)
            }
            completion(response)
        }
    }
}
```
Summary

Message Filter extensions

Powerful but subject to privacy restrictions

Try making one!
More Information

https://developer.apple.com/wwdc17/249
<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy and Your Apps</td>
<td>Executive Ballroom</td>
<td>Tuesday 11:20AM</td>
</tr>
<tr>
<td>What’s New in Foundation</td>
<td>Hall 2</td>
<td>Wednesday 11:00AM</td>
</tr>
<tr>
<td>Seamless Linking to Your App</td>
<td>WWDC15</td>
<td></td>
</tr>
<tr>
<td>CallKit and Identity Lookup Lab</td>
<td>Technology Lab H</td>
<td>Tuesday 12:40PM–2:50PM</td>
</tr>
</tbody>
</table>