The Module/Class Name: Message

In the swarms.agents framework, the class `Message` is used to represent a message with timestamp and optional metadata.

Overview and Introduction

The `Message` class is a fundamental component that enables the representation of messages within an agent system. Messages contain essential information such as the sender, content, timestamp, and optional metadata.

Class Definition

Constructor: `__init__`

The constructor of the `Message` class takes three parameters:

1. `sender` (str): The sender of the message.
2. `content` (str): The content of the message.
3. `metadata` (dict or None): Optional metadata associated with the message.

Methods

1. `__repr__(self)`: Returns a string representation of the `Message` object, including the timestamp, sender, and content.

```python
class Message:
    """
    Represents a message with timestamp and optional metadata.

    Usage
    ----------
    mes = Message(
        sender = "Kye",
        content = "message"
    )

    print(mes)
    """

    def __init__(self, sender, content, metadata=None):
        self.timestamp = datetime.datetime.now()
        self.sender = sender
        self.content = content
        self.metadata = metadata or {}

    def __repr__(self):
        """
        __repr__ represents the string representation of the Message object.

        Returns:
```

        (str) A string containing the timestamp, sender, and content of the message.
        """
        return f"{self.timestamp} - {self.sender}: {self.content}"
```

## Functionality and Usage

The `Message` class represents a message in the agent system. Upon initialization, the `timestamp` is set to the current date and time, and the `metadata` is set to an empty dictionary if no metadata is provided.

### Usage Example 1

Creating a `Message` object and displaying its string representation.

```python
mes = Message(sender="Kye", content="Hello! How are you?")

print(mes)
```

Output:
```
2023-09-20 13:45:00 - Kye: Hello! How are you?
```
```

Usage Example 2

Creating a `Message` object with metadata.

```python
metadata = {"priority": "high", "category": "urgent"}
mes_with_metadata = Message(
    sender="Alice", content="Important update", metadata=metadata
)

print(mes_with_metadata)
```

Output:
```
2023-09-20 13:46:00 - Alice: Important update
```

Usage Example 3

Creating a `Message` object without providing metadata.

```python
mes_no_metadata = Message(sender="Bob", content="Reminder: Meeting at 2PM")

print(mes_no_metadata)
```

```

Output:

```
2023-09-20 13:47:00 - Bob: Reminder: Meeting at 2PM
```

## Additional Information and Tips

When creating a new `Message` object, ensure that the required parameters `sender` and `content` are provided. The `timestamp` will automatically be assigned the current date and time. Optional `metadata` can be included to provide additional context or information associated with the message.

## References and Resources

For further information on the `Message` class and its usage, refer to the official swarms.agents documentation and relevant tutorials related to message handling and communication within the agent system.
```